Inland Revenue
Te Tari Taake

Inland Revenue

Identity and Access Services build pack

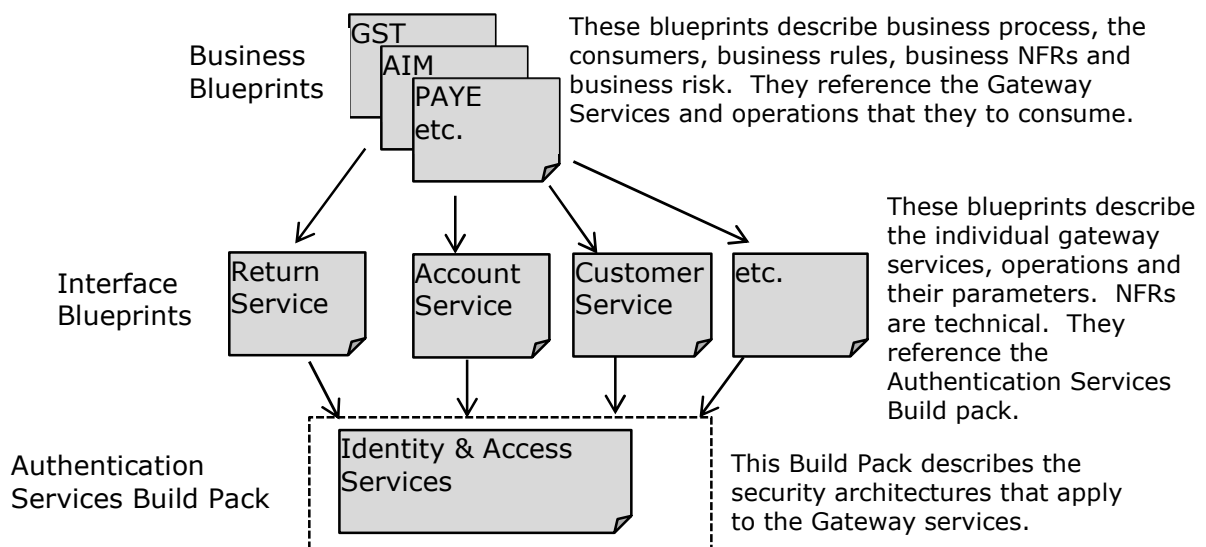**Date:** 1/12/2020
Version: 2.8

Inland Revenue
Te Tari Taake

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

Inland Revenue
Te Tari Taake

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.

The 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:

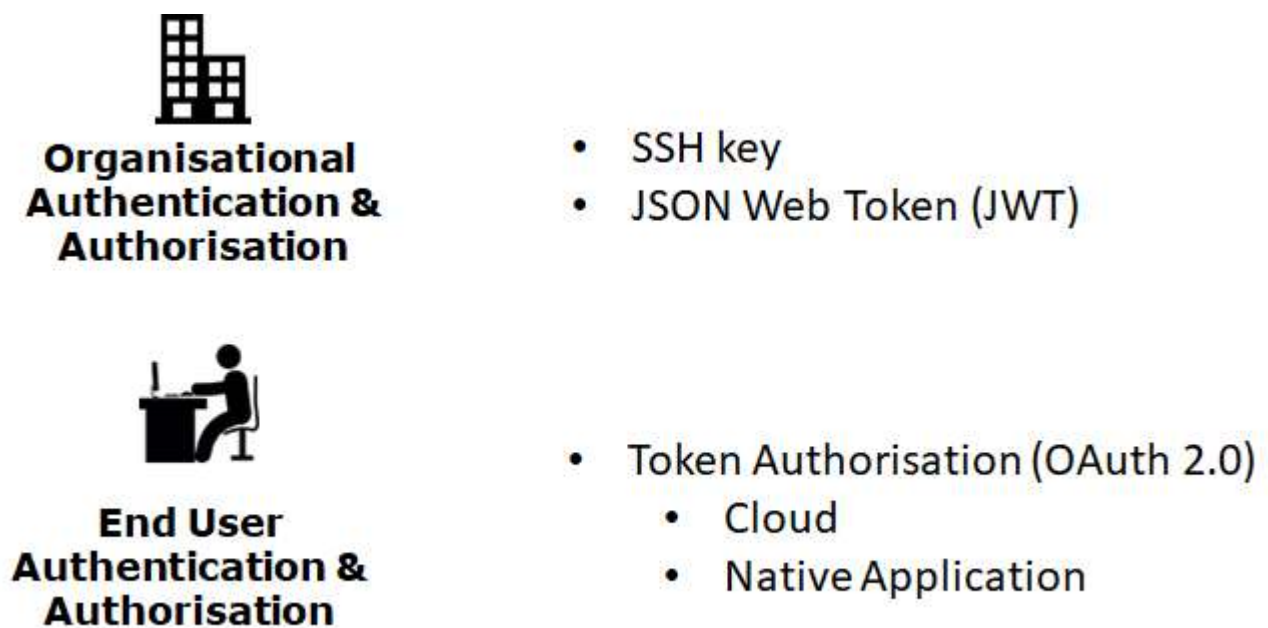**Organisational Authentication & Authorisation**

- SSH key
- JSON Web Token (JWT)

End User Authentication & Authorisation

- Token Authorisation (OAuth 2.0)
  - Cloud
  - Native Application

Figure 1 Types of Entity and Authentication & Authorisation mechanisms

Inland Revenue
Te Tari Taake

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
SSH keys	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.
JSON Web Token (JWT) **Using JSON Web Signature (JWS) for digital signatures with RSA/ECDSA**	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 3rd 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
Cloud application Token Auth **OAuth2.0 based**	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.
Native application Token Auth **OAuth2.0 based**	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.

![Inland Revenue Te Tari Taake logo]

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:
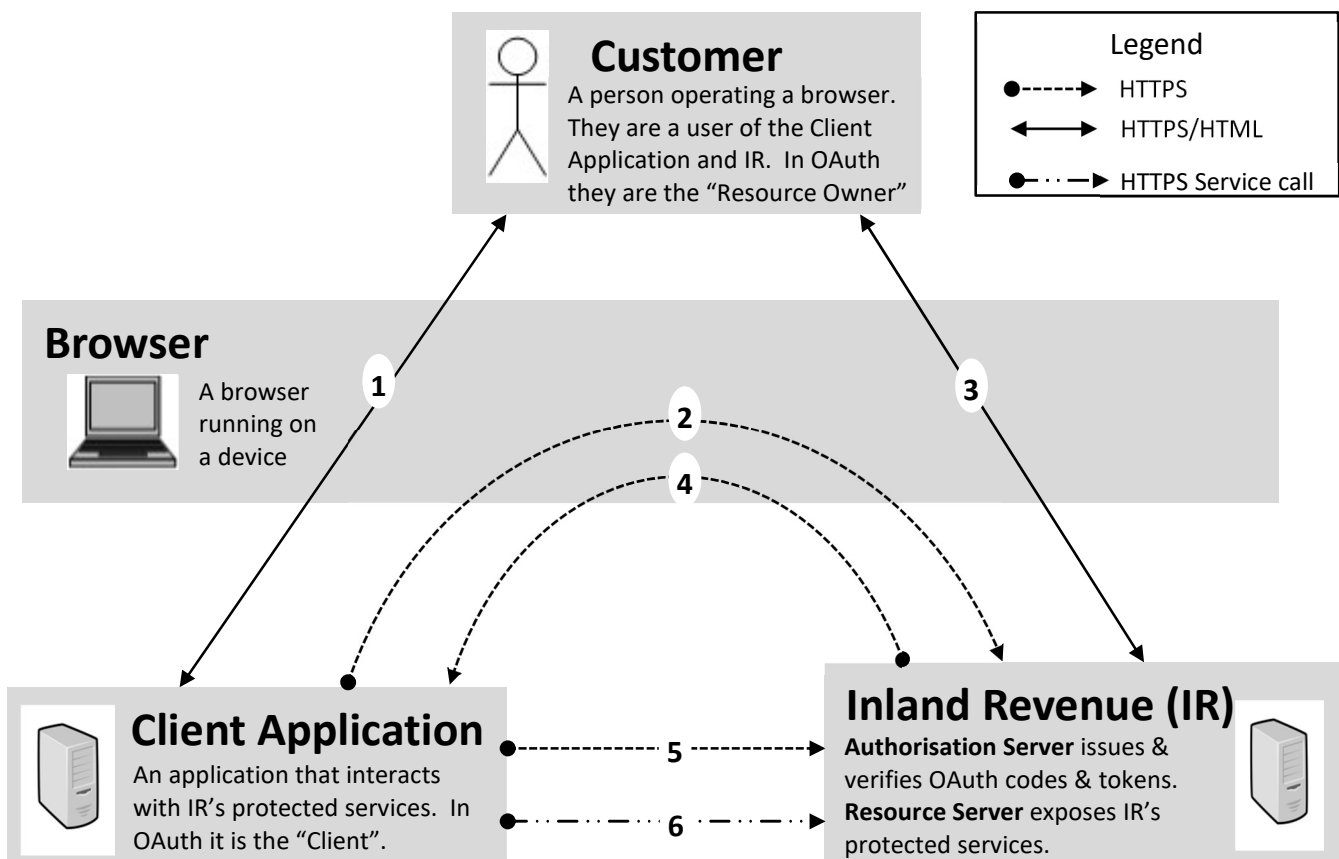
1. The User is interacting with the Client Application. They access a protected service provided by IR (e.g. to file a return, retrieve a balance etc.)
2. The Client Application invokes the Authorisation API to get an authorisation code, the user's browser is redirected to IR's logon page.
3. IR prompts 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

Query parameters must be added in the POST body. Query parameters in the POST request will not be accepted.

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/authori
ze?

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
response_type	Response type requested	Required	"code"

client_id	The agreed Client identifier established at registration. Inland Revenue maintains this list of values.	Required	client_id
redirect_uri	The Client application's redirect URI to which the Authorisation Code is returned.	Required	Business Partner defined & agreed with IR
scope	Use space-separated values. Define scope values in the configuration/scope registry.	Required	"MYIR.Services"
state	A value used by the Business Partner to maintain state between the request and callback. 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 https://tools.ietf.org/html/rfc6749 for more information (Section 12.12).	Recommended	Business Partner defined. Must be < 200 characters. Allowed Characters: a-z A-Z 0-9 - . ? , : / \ + = $ # Not space (%20, (ASCII 32)
logout	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. This will force the re-display of the login page and request the input of a new myIR account to login. The value of "false" or absence of this parameter will not force logout.	Optional	"true" "false"

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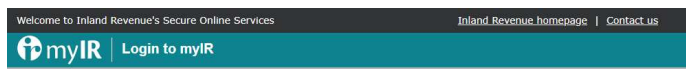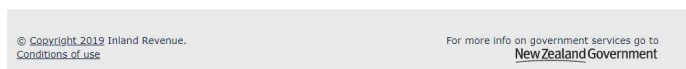
If the software provider chooses (not generally recommended as this page may change from time to time) to present this page within a frame the minimum recommended size in pixels is 600w x 500h.

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.

Please contact GatewayServices@ird.govt.nz for revoke consent assistance.

Inland Revenue
Te Tari Taake

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
Code	Authorisation code value - Includes the following: • Expiry • Client_id • Redirect_uri	Encrypted string ~1000 characters
State	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 https://tools.ietf.org/html/rfc6749 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
400	**invalid_redirect_uri**	Redirect URI mismatch with Business Partner app
401	**Invalid client ID**	API Key contains invalid information
	invalid_client	Business partner identifier invalid
	invalid_scope	Requested scope is invalid, unknown, or malformed
	server_error	Authentication - Runtime processing error
	access_denied	End-user denied authorisation
500	**InternalError**	An internal and unexpected error occurred
504	**GatewayError**	Gateway did not receive a timely response from the upstream server

2.1.3.6   Request Access 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
redirect_uri	The Client application's redirect URI for the Authorisation Token.	Required	Business Partner defined & agreed with IR
grant_type	The grant type is authorization_code	Required	authorization_code
code	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            NTQzMjFpZZ...ZWxjb21lMQ==

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

With the values of:

Name	Description	Required	Valid Values
Authorization	"Basic" + Base64 encoded (ClientID + ":" + Client Secret)	Required	Base64 encoded string
Content-Type	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:

When an error occurs, a corresponding HTTP Status code is returned. In addition, a JSON payload is return in the response body.

The error response should not be used within client code to determine the course of action. The error messages are informational, intended to be logged for human review during integrating with IRD.

The structure of the error response body is:

`{"error": "Type of error", "error_description": "Description of the error"}`

HTTP code	Error Type	Description
400	`unsupported_grant_type`	Invalid/unsupported value for `grant_type`
		Grant Type not `authorization_code`, or unrecognised parameters in the request
	`invalid_request`	Invalid Client
		Authorisation Header is incorrect
401	`Invalid_grant`	Invalid Redirect URI
		Redirect URI mismatch with Business Partner app
	`invalid_grant`	Invalid AuthZ Code
		AuthZ Code expired, used multiple times, or invalid.
		Client should retry the Authorisation service to get a new AuthZ Code
500	`InternalError`	An internal and unexpected error occurred
504	`GatewayError`	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. Note the Refresh token is no longer a JWT.

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
grant_type	The grant type is required and takes the value refresh_token	Required	refresh_token
refresh_token	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   NTQzMjFpcZ...ZWxjb21lMQ==

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

With the values of:

Name	Description	Required	Valid Values
Authorization	"Basic " + Base64 encoded (ClientID + ":" + Client Secret)	Required	Base64 encoded string
Content-Type	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.

Errors:

HTTP code	Error Type	Description
400	unsupported_grant_type	Invalid/unsupported value for grant_type. Grant Type not refresh_token, or unrecognised parameters in the request
	invalid_request	Invalid Client Authorisation Header is incorrect
401	Invalid_grant	This client is not granted with some or all of requested scope(s)

		The underlying consent has been revoked by the user, invalidating the refresh token. The client will need to re-authorise with the user to get consent.
	`invalid_grant`	Invalid Refresh Token Refresh token is expired or invalid. Client should check the value of the refresh_token. If no errors here, the client should re-authorise the user and get a new refresh token.
500	`InternalError`	An internal and unexpected error occurred
504	`GatewayError`	Gateway did not receive a timely response from the upstream server

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=startLogon exp

</form>
```

With the values of:

Name	Description	Required	Valid Values
grant_type	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
Oracle_token_action	The oracle token action is required and	Required	validate

	takes the value validate		
Scope	Scope is required and takes the value MYIR.services	Required	MYIR.services
assertion	The assertion field contains the Access Token	Required	<Access Token>
oracle_token_attrs_retrieval	This field details the list of fields to retrieve. startLogon is user ID Exp is expiry time in unix epoch	Required	startLogon exp

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

Authorization: Basic   NTQzMjFFpZ...ZWxjb21lMQ==

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

With the values of:

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

Errors:

HTTP code	Error Type	Description
400	unsupported_grant_type	Invalid/unsupported value for grant_type. Grant Type not refresh_token, or unrecognised parameters in the request
401	Invalid_grant	Access Token Validation Failed Access token is expired or not valid. The client should re-authorise or use the refresh token to get a valid request

403	unauthorized_client"	This client is not granted with some or all of requested scope(s)
		The access token is no longer valid as the user has revoked consent. Clients should re-authorize the user.
500	InternalError	An internal and unexpected error occurred
504	GatewayError	Gateway did not receive a timely response from the upstream server

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

<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
grant_type	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
Oracle_token_action	The oracle token action is required and takes the value delete	Required	delete
assertion	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	NTQzMjFpFpZ...ZWxjb21lMQ==
Content-Type: application/x-www-form-urlencoded;charset=UTF-8	

With the values of:

Name	Description	Required	Valid Values
Authorization	"Basic" + Base64 encoded (ClientID + ":" + Client Secret)	Required	Base64 encoded string
Content-Type	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.

The OAuth Refresh token does not expire.

Note, each new logon by a user will result in a new OAuth and Refresh token set. Service Providers need to ensure adequate care is taken when storing these tokens.

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

Token: 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

Token: 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. Refresh token functionality is not available for Native App OAuth partners.

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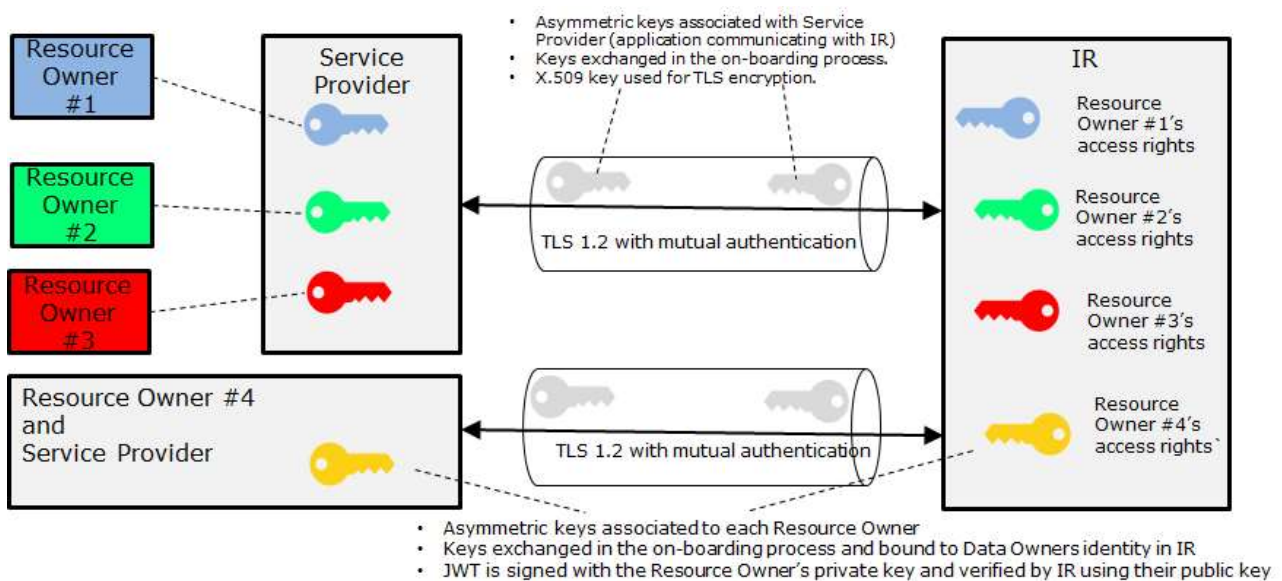
As a result, IR will know who the token comes from (the Resource Owner), and from interrogating the token will also know the myIR logon from the startLogon value. IR will use this myIR logon information to apply the delegations' model already existing in eServices.

Where startLogon is null, then the Resource Owner's relationships inherent in START will be used to determine the information available to the calling service. This is done by mapping the Client Signed certificate thumbprint to the Customer in START.

The hierarchy of use is the startLogon and then Customer. If the start_Logon is populated, then the delegations of the myIR logon will be used. If the start_Logon field is not populated, then the Customer delegations in START will be used.

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

Inland Revenue
Te Tari Taake

Using mutual TLS will enable IR to 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
sub	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
alg	Signature or encryption algorithm.	Required	RS256, RS384, RS512 ES256, ES384, ES512
typ	Type of token.	Required	JWT
kid	Key ID.	Required	M2M

iss	Issuer who created this token.	Required	Value agreed between issuer and IR e.g. www.name.com
startLogon	The myIR logon of a representative of the token subject. The Subject must be the data owner.	Optional	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 Data Owner's relationships determines the delegation rights for the calling service. Refer to individual Service Build packs to understand when this applies.
iat	Issued at (seconds since Unix epoch).	Required	Must not precede the signing certificate issue date e.g. 1560144847
exp	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
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS1-v1_5 using SHA-512	Optional
ES256	ECDSA using P-256 and SHA-256	Recommended+
ES384	ECDSA using P-384 and SHA-384	Optional
ES512	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 on be on a secure, trusted host, and the issuer must always protect their private key.

- 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.

3 Appendix A – Sample payloads

3.1 Request Authorisation Code

Service consumer to IR. Refer to step 2 in section 2.1.1.

3.1.1 Request

Service provider will send a HTTP GET request to Auth Server, url format as below:

https://test4.services.ird.govt.nz/ms_oauth/oauth2/endpoints/oauthservice/authorize?client_id=Test99999999&redirect_uri=https://myreturnuri/test/&scope=MYIR.Services&response_type=code&state=123&logout=true

3.2 Authorisation Code response

IR to Service consumer. Refer to step 4 in section 2.1.1.

3.2.1 Success Response – Authorisation Code sent

https://myreturnuri/test/?code=b2poZWt3RUZ4eXU3UnJmZjhHYkRxUT09fnl3SkJuR2FnWDhTd
VlSNEkvR0hhejcyUU55VTc2bGJmODgxcmROUXY3UXV0RG0zaURdNTUc2T2NjOHRrb3I5Mm1YSE1kMHFtN
202VFRFejlLUXJaa2dGMFJSNUZDTDZzNVZKeHhoRGxoWEd5cGhCV2lYV1llUDVxUVFaK2I4STAxTEh2W
WE0MVU1bmE0TFZORUd1QzJuVC9qVys1T2J5RjEvWmQ5OVpmY3hWU3B3ZUcrdk5Ka2ttbFhVNExmMG1wb
E5KYlZMd2d2dTU1k4SVNFVDV4MStJdm04d2tQOW1hQVE2b2ZRY2lSZVAzQW0xNkFsbkh6eHdNNitCaFpzW
U55ZzJna3ZSb095WkRSeERRKMWFlNTRzV3lSTGJhWU1Sb1FOVFdTdRUk1JNXdUZWpFNnhsVmc1ZEd2QUxrW
ld0QmhpcTlvTkVRWlB2anRubFJ39XUFhpRnFFIQ0dwWjDbmRaY0FLQWFJYy9ScFRsQ0pLSFFFRRjduZjV
LU3ZtVXc4TVFsbWsrNlI5V2d6RGxpNW5DandpR2lrK0l0UEY4SzRPMjFRRRTZLNnAydmVnZW54S0ZQOS9
WZFJEMG9OcXlLdC9Oc1h4TlFLYUR0YzFXMXpem1HcXNNUVUw2ZGM5cWlDMWdHdHllYK0pZWk5EbFhRdVI
0RFVvWmRTd0tzVVhkVOE9VRVUyaG13a2FQb1kybnlZb2pBYkkg2QkdQYUpIdmtrTkxxEcVdRbVp0Zm8yaEM
2UkZhVzBDUW1uN1pUMVZmmS3grcXhmmMDVST29kRW16Y3c0SEdheUFFRYWViUUE9PD==&state=123

3.3 Request Access token

Service consumer to IR. Refer to step 5 in section 2.1.1.

3.3.1 Exchange Authorisation Code for oAuth Access Token.

At this step as a service provider you need to send a HTTP POST request to Auth Server with Authorization Code and your client credentials.
Url format:

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

Request headers: {Authorization=[Basic
VGVzdDMwMMjA2NDkyOk9hdXRoMklSU2VjcmV0dA==], Content-Type=[application/x-www-form-
urlencoded], Content-Length=[1127]}

Request body:
redirect_uri=https://myreturnuri/test/&grant_type=authorization_code&code=eyJhbG
ciOiJSUzUxMiIsInR5cCI6IkpXVCIsIng1dCI6Ino5VFR1R2pHSVJd01pQjRaUGxjbzhDZ3JlOCIsIm
tpZCI6Im9yYWtleSJ9.eyJvcmFjbGUub2F1dGgucmVkaXJlY3QtdXJpIjoiaHR0cHM6Ly9qZW5raW5zL
mtppd2lzYXZlci5pcmQuQuZGlnaXRhbBhcnRuZXIuc2VydmljZXMvIiwic3ViIjpudWxsLCJvcmFjbGUub
2F1dGgudXNlcl9vcmlnaW5faWRfdHlwZSI6IkxEQVBfVUlEIiwib3JhY2xlLm9hdXRoLnVzZXJfb3JpZ
2luX2lkIjoic2FtbXkzOTAiLCJpc3MiOiJJbmxhbmRSZXZlbnVlIiwib3JhY2xlLm9hdXRoLnN2Y19wX
24iOiJPQXV0aFNlcnZpY2Vjcm9maWxlIiwiaWF0IjoxNTAyMzIxNDI3LCJvcmFjbGUub2F1dGgudGtfY
29udGV4dCI6ImF6YyIsImV4cCI6MTUwMjM1MDIyNywicHJuIjpudWxsLCJqdGkiOiJhNDg2ZTU1Y0zZ
Tc1LTQ3ZmYtODk0NC1hNTcxZWNlYzgzNmYiLCJvcmFjbGUub2F1dGguc2NvcGUiOiJNWUlSLllNlcnZpY
2VzIiwib3JhY2xlLm9hdXRoLmNsaWVudF9vcmlnaW5faWQiOiJUZXN0MzAyMDY0OTIiLCJ1c2VyLnRlb

mFudC5uYW1lIjoiRGVmYXVsdERvbWFpbiIsIm9yYWNsZS5vYXV0aC5pZF9kX2lkIjoiMTIzNDU2NzgtM
TIzNC0xMjM0LTEyMzQtMTIzNDU2Nzg5MDEyIn0.NTBu3R-JwaaOWfvMdWAHqY7Ji3YI3I-
bSTXqx6jauqEUhswLmAG6cbpGaSky50ECbHNv2skU8WVZ0RYv67KPgITGXJz0ZKSjqOgiZ0R4kFCZ7as
N8yjIzXgxwWk4mPXL5E02u24-VMbr_hrNZYDZbakOpz4uY6UlSSNECmw0ac8

3.3.2 Success Response – Access Token sent

The response contains the Access Token and expiry time. If refresh tokens are used this is also returned:

```
{
    "expires_in": 28800,

    "token_type": "Bearer",

    "access_token":
"eyJhbGciOiJSUzUxMiIsInR5cCI6IkpXVCIsIng1dCI6Ino5VFR1R2pHSVJNd01pQjRaUGxjbzhDZ3J
lOCIsImtpZCI6Im9yYWtleSJ9.eyJzdWIiOiJUUjI0NTczNzczIiwib3JhY2xlLm9hdXRoLnVzZXJfb3
JpZ2luX2lkX3R5cGUiOiJMREFQX1VJRCIsIm9yYWNsZS5vYXV0aC51c2VyX29yaWdpbl9pZCI6IlRSMj
Q1NzM3NzMiLCJpc3MiOiJJBmxhbmRSZXZlbnVlIiwib3JhY2xlLm9hdXRoLnN2Y19wX24iOiJPQVV0aF
NlcnZpY2VQcm9maWxlIiwiaWF0IjoxNTEzMTEzNDk4LCJvcmFjbGUub2F1dGguHcuLmlkX3R5cGUiOi
JMREFQX1VJRCIsIm9yYWNsZS5vYXV0aC50a19jb250ZXh0IjoicmVzb3VyY2VfYWNjZXNzX3RrIiwiZX
hwIjoxNTEzMTQyMjk4LCJwcm4iOiJUUjI0NTczNzczIiwianRpIjoiZDRlMDdkMzYtNDFlMi00OTg5LT
kxNGMtNDYwZmUzNmYwZGQ0Iiwib3JhY2xlLm9hdXRoLmNsaWVudF9vcmlnaW5faWQiOiJNWU9CMTI3NT
QyMSIsIm9yYWNsZS5vYXV0aC5zY29wZSI6Ik1ZSVIuU2VydmljZXMiLCJ1c2VyLnRlbmFudC5uYW1lIj
oiRGVmYXVsdERvbWFpbiIsIm9yYWNsZS5vYXV0aC5pZF9kX2lkIjoiMTIzNDU2NzgtMTIzNC0xMjM0LT
EyMzQtMTIzNDU2Nzg5MDEyIn0.P4c0twW-MFrGFuKa8gzuEC0AwxhMJDHSge5L7fUkWKF-
9k3WCmIwu5sKSxq2MNiLxs1XBDboqS8RA4-prv85-
sWQLN0patssDzK2qLCgWa9lKyTExIpmP0CvV_Yrnf5KCTNjyNzRl5Ba30T7bDaz5gn5iivrJn1HNghhb
horY1I"
```

The response may contain the Refresh Token. Note this is no longer a JWT.

```
    "refresh_token":
"yI0nEXXc13huo6V1flV4vg%3D%3D%7E5GWEUoj3Y6GwPs69GPZU92TFFhZ2qrLezRiiY7K7XMFExmCH
GcvM6BmeKzu6Tgj5sD7vCE%2BY22Boy7yRRet6mvvk4P4J0mSa2P7omd4G%2F08IoqsUgz4HmMe2%2FL
VeTN1bWXRxaaa6fHbRryCHJ2p6Raj2RB8cbwfDCRHhI5o2logjGZbIozUQL78xX6PiQk9PYh8%2FHU2R
2IkV9SOWv3EO0XutuzYbJ0X4gAb9X0qW0iALvHnxk8bwmrPoLOYu%2Ba4ZRe5uw5doaAxANikjhHnhjb
JLt%2FyzGLn056jdmcydjXw%3D",

}
```

3.4 Request Refresh token

Service consumer to IR. Refer to section 2.1.4.

3.4.1 Refresh request

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

```
Request headers: {Authorization=[Basic
VGVzdDMwMjA2NDkyOk9hdXRoMklSU2VjcmV0dA==], Content-Type=[application/x-www-form-
urlencoded], Content-Length=[1127]}
```

```
Request body:
grant_type=refresh_token&refresh_token=yI0nEXXc13huo6V1flV4vg%3D%3D%7E5GWEUoj3Y6
GwPs69GPZU92TFFhZ2qrLezRiiY7K7XMFExmCHGcvM6BmeKzu6Tgj5sD7vCE%2BY22Boy7yRRet6mvvk
4P4J0mSa2P7omd4G%2F08IoqsUgz4HmMe2%2FLVeTN1bWXRxaaa6fHbRryCHJ2p6Raj2RB8cbwfDCRHh
I5o2logjGZbIozUQL78xX6PiQk9PYh8%2FHU2R2IkV9SOWv3EO0XutuzYbJ0X4gAb9X0qW0iALvHnxk8
bwmrPoLOYu%2Ba4ZRe5uw5doaAxANikjhHnhjbJLt%2FyzGLn056jdmcydjXw%3D
```

3.4.2   Refresh token reply

The Refresh token reply is the same as an authorisation code exchange for an Access token.

```
{

    "access_token":
"eyJhbGciOiJSUzUxMiIsInR5cCI6IkpXVCIsIng1dCI6Ino5VFR1R2pHSVJNd01pQjRaUGxjbzhDZ3J
lOCIsImtpZCI6Im9yYWtleSJ9.eyJzdWIiOiJUUjI0NTczNzczIiwib3JhY2xlLm9hdXRoLnZzZXfb3
JpZ2luX2lkX3R5cGUiOiJMREFQX1VJRCIsIm9yYWNsZS5vYXV0aC51c2VyX29yaWdpbl9pZCI6IlRSMj
Q1NzM3NzMiLCJpc3MiOiJJbmxhbmRSZXZlbnVlIiwib3JhY2xlLm9hdXRoLnN7Y19wX24iOiJPQXV0aF
NlcnZpY2VQcm9maWxlIiwiaWF0IjoxNTEzMTE0NzY4LCJvcmFjbGUub2F1dGguHJuLmlkX3R5cGUiOi
JMREFQX1VJRCIsIm9yYWNsZS5vYXV0aC50a19jb250ZXh0IjoicmVzb3VyY2VfYWNjZXNzX3RrIiwiZX
hwIjoxNTEzMTQzNTY4LCJwcm4iOiJUUjI0NTczNzczIiwianRpIjoiMTkzNWEyZjktZGYwNi00NzA1LW
I5MDMtYmQ1ZTY5NzIwZjE2Iiwib3JhY2xlLm9hdXRoLmNsaWVudF9vcmlnaW5faWQiOiJNWU9CMTI3NT
QyMSIsIm9yYWNsZS5vYXV0aC5zZ29wZSI6Ik1ZSVuU2VydmljZXMiLCJ1c2VyLnRlbmFudC5uYW1lIj
oiRGVmYXVsdERvbWFpbiIsIm9yYWNsZS5vYXV0aC5pZF9kX2lkIjoiMTIzNDU2NzgtMTIzNC0xMjM0LT
EyMzQtMTIzNDU2Nzg5MDEyIn0.vMo0b_H8ZeC09xcfDHoyRl50eqk0GeYdake99NrIvzcxUCsRtHbwMm
gnBvzy45yL6T07-jfBlgRx3Tzt0MP-Gc7OKx-
ma_s2SCD39m_q1by9nX_x7BWKFNGMM9gJl7jYTF6sWP6nNJCJWXfilfHJVuRchenyW4Lv8sVTKGPbQOQ
",

    "token_type": "Bearer",

    "expires_in": 28800,

    "refresh_token":
"yI0nEXXc13huo6V1flV4vg%3D%3D%7E5GWEUoj3Y6GwPs69GPZU92TFFhZ2qrLezRiiY7K7XMFExmCH
GcvM6BmeKzu6Tgj5sD7vCE%2BY22Boy7yRRet6mvvk4P4J0mSa2P7omd4G%2F08IoqsUgz4HmMe2%2FL
VeTN1bWXRxaaa6fHbRryCHJ2p6Raj2RB8cbwfDCRHhI5o2logjGZbIozUQL78xX6PiQk9PYh8%2FHU2R
2IkV9SOWv3EO0XutuzYbJ0X4gAb9X0qW0iALvHnxk8bwmrPoLOYu%2Ba4ZRe5uw5doaAxANikjhHnhjb
JLt%2FyzGLn056jdmcydjXw%3D"

}
```

3.4.3   Error Response.

See section 2.1.3.6 for a more detailed list of error responses, a typical JSON response:

```
{

    "error": "invalid_grant",

    "error_description": "Invalid Grant: grant_type=refresh_token"

}
```

3.5   Validate token request

Service consumer to IR. Refer to section 2.1.5

3.5.1   Validate token request

As a service provider you need to send a HTTP POST request to Auth Server with Access Token to validate your client credentials.

Url format:

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

```
Request headers: {Authorization=[ Basic
VGVzdDMwMjA2NDkyOk9hdXRoMklSU2VjcmV0dA==], Content-Type=[application/x-www-form-
urlencoded], Content-Length=[1127]}
```

Inland Revenue
Te Tari Taake

Request body: grant_type=oracle-idm:/oauth/grant-type/resource-access-token/jwt&oracle_token_action=validate&scope=MYIR.Services&assertion=eyJhbGciOiJSUzUxMiIsInR5cCI6IkpXVCIsIng1dCI6Ino5VFR1R2pHSVJNd01pQjRaUGxjbzhDZ3JlOCIsImtpZCI6Im9yYWtleSJ9.eyJzdWIiOiJDbGllbnQwMDgiLCJvcmFjbGUub2F1dGgudXNlcl9vcmlnaW5faWQiOiJDbGllbnQwMDgiIiwiaXNzIjoiSW5sYW5kUmV2ZW51ZSIsIm9yYWNsZS5vYXV0aC5zdmNfcF9uIjoiT0F1dGhTZXJ2aWNlUHJvZmlsZSIsImlhdCI6MTUxOTM2OTExMCwib3JhY2xlLm9hdXRoLnByb2ZpbGVfdHlwZSI6IlRERUF9VSUQiLCJvcmFjbGUub2F1dGgudGtfY29udGV4dCI6InJlc291cmNlX2FjY2Vzc190a2oiLCJleHAiOjE1MTkzNkNrxMCwicHJuIjoiQ2xpZW50MDA4IiwianRpIjoiZjgwOTEzNGEtZTE3Ny00OGMwLWJjNDQtMzJmMDg3NDQxMTUxIiwib3JhY2xlLm9hdXRoLnNsaWVudF9vcmlnaW5faWQiOiJUZXN0MzAyMDY0OTIiLCJvcmFjbGUub2F1dGguc2NvcGUiOiJNWUlSLlNlcnZpY2VzIiwidXNlci50ZW5hbnQubmFtZSI6IkRlZmF1bHREb21haW4iLCJvcmFjbGUub2F1dGguaWRfZF9pZCI6IjEyMzQ1Njc4LTEyMzQtMTIzNC0xMjM0LTEyMzQ1Njc4OTAxMiJ9.3PeIiEFSftcY482O74TjEDovYObjbEcjF_rqH8JMmqJgLLiBV4oDxkj3hlak4jX4NuG-mpLw7E2BYiFfuS2LtRo9R1oko-CO9F5JALxMD5MVxtb2KQFxrhiF-n1PFZhJzQydcHOkmBBxZDDN6OciS4SfeEm6s1iN46QldbunwYM&oracle_token_attrs_retrieval=startLogon exp

3.5.2 Validate token reply

A sample response payload:

```
{
    "successful": true,
    "oracle_token_attrs_retrieval":    {
        "exp": 1519367910,
        "startlogon": "Client008"
    }
}
```

3.5.3 Validate token Error Response.

Sample payloads when there is an error:

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

3.6 Revoke token request

Service consumer to IR. Refer to section 2.1.6.

3.6.1 Revoke token request

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.

Url format:

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

Request headers: {Authorization=[Basic
VGVzdDMwMjA2NDkyOk9hdXRoMklSU2VjcmV0dA==], Content-Type=[application/x-www-form-
urlencoded], Content-Length=[1127]}

Request body: grant_type= oracle-idm:/oauth/grant-type/resource-access-
token/jwt&oracle_token_action=delete&assertion=eyJhbGciOiJSUzUxMiIsInR5cCI6IkpXV
CIsIng1dCI6Ino5VFR1R2pHSVJNd01pQjRaUGxjbzhDZ3JlOCIsImtpZCI6Im9yYWtleSJ9.eyJzdWIi
Om51bGwsIm9yYWNsZS5vYXV0aC51c2VyX29yaWdpbl9pZF90eXBlIjoiTERBUF9VSUQiLCJvcmFjbGUu
b2F1dGgudXNlcl9vcmlnaW5faWQiOiJUUUjI0NTczNzczIiwiaXNzIjoiSW5yYW5kUmV2ZW51ZSIsIm9y
YWNsZS5vYXV0aC5ydC50dGMiOiJyZXNvdXJjZV9hY2Nlc3NfdGsiLCJvcmFjbGUub2F1dGguc3ZjX3Bf
biI6Ik9BdXRoU2VydmljZVByb2ZpbGUiLCJpYXQiOjE1MTMxMTM0OTgsIm9yYWNsZS5vYXV0aC50a19j
b250ZXh0IjoicmVmcmVzaF90b2tlbiIsImV4cCI6MTUxMzE0MjI5OCwicHJuIjpudWxsLCJqdGkiOiJm
NDZmMTU1YS1jNGU4LTRkYTktOTFkMy0zMjUxZTIyNDNhNjgiLCJvcmFjbGUub2F1dGguc2NvcGUiOiJN
WUlSLlNlcnZpY2VzIiwib3JhY2xlLm9hdXRoLmNsaWVudF9vcmlnaW5faWQiOiJNWU9CMTI3NTQyMSIs
InVzZXIudGVuYW50Lm5hbWUiOiJEZWZhdWx0RG9tYWluIiwib3JhY2xlLm9hdXRoLmlkX3RfaWQiOiIx
MjM0NTY3OC0xMjM0LTEyMzQtMTIzNC0xMjM0NTY3ODkwMTIifQ.bsqzeC1BWEAireUWCLq3ZGhbSjcb6
jrkfbV7F8lLuujVMF7xlGkOMoSLx9DSN_ePPE4z7PeDbz3igF269siiuplkitbA6JZiR6DRs3pl9ED3M
Cjwzp4_DVxXuip0K0FOamwOwzyROzSupjYsEEkJhkG0ur2SGdTnjHFyXFoleoU

3.6.2 Revoke token reply

A sample response payload:

{

"successful":true

}

3.6.3 Revoke token Error Response.

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
Client Application	A Client Application is an operating instance of Software that is deployed in one or more sites. A number of deployment patterns are possible:

	• A single cloud-based instance with multiple tenants and online users, • An on-premise instance (e.g. an organisation's payroll system) • A desktop application with an online user.
Customer	A Customer is the party who is a taxpayer 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. 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.
Intermediary	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.
Mutual authentication	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). Mutual TLS can be referred to as mTLS
OAuth 2.0	OAuth 2.0 is an industry-standard protocol for authorization
Native app	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.
Protected Service	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.
Resource Owner	Legal owner of the data or resources user in message exchanges with Inland Revenue
SFTP	Secure File Transport Protocol
Software	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.
Software Developer	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.
Service Provider	Entity running an application communicating with Inland Revenue

Inland Revenue
Te Tari Taake

TLS 1.2	A cryptographic protocol that provides communications security over a computer network. Version 1.2 is mandated in most cases.
WS-Security	An extension to SOAP to apply security to Web Services. An OASIS Web service specification
X.509 Certificate	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
'ew0KCSJhbGciOiAiUlMyNTYiLCANCgkidHlwIjogIkpXVCIsIA0KCSJraWQiOiAiTTJNIg0KfQ.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
'ew0KCSJhbGciOiAiUlMyNTYiLCANCgkidHlwIjogIkpXVCIsIA0KCSJraWQiOiAiTTJNIg0KfQ.ew0KIC
Aic3ViIjogIldpZ2V0IENvbXBhbnkiLA0KICAiaXNzIjogInd3dy53aWdldGNvbXBhbnkuY29tIiwNCiAg
InN0YXJ0TG9nb24iOiAibXlJUndlYmxvZ2luVXNlciIsDQogICJpYXQiOiAxNTYwODk0NzMzLA0KICAi
ZXhwIjogMTU2MDkwNDczMw0KfQ' | openssl dgst -sha256 -sign jwtRS256.key -binary |
openssl base64 -A

The base64 signature needs to be made base64url encoded, which requires character substitution:

- '+' translates to '–'

Inland Revenue
Te Tari Taake

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

JWT Token:

ew0KCSJhbGciOiAiUlMyNTYiLCANCgkidHlwIjogIkpXVCIsIA0KCSJraWQiOiAiTTJNIg0KfQ.ew0KIC
Aic3ViIjogIldpZ2V0IENvbXBhbnkiLA0KICAiaXNzIjogInd3dy53aWdldGNvbXBhbnkuY29tIiwNCiАg
InN0YXJ0TG9nb24iOiAibXlJUndlYmxvZ2luVXNlciIsDQogICJpYXQiOiAxNTYwODk0NzMzLA0KICAi
ZXhwIjogMTU2MDkwNDczMw0KfQ.R4ht3N7jsMSRaeXAVSHtysNeymBexO-
0okop79EYVFqgvOZduvChyMIXkhxSX_7HhQNKcJhtdOL64e0EPUiElBpqS7GpA9cf8p_ktoweu6ge
kjeVPSNHoDokYPbSi4y8NblOdSdr1ECKuSjsDvekc6vYN7f0j4Rrn3sr7Y-
0hpWEQgCiTcbUIfEHGm1nQHX9mFxDeJIYgyG4XDQXB1rF-LAIL_omDWGoRBBVyXSQk-
zb1uHO8yhJv11j955Tz4GQD86DVvmgZPKoftPG73BGoyRzWaYH9vCi8AIZTRnAEUqwWvrw81MU
up1V6775LOcy83OnOka3tLSZ_SnbctDSiA

-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsLIJZl5ePxnDzoPChQ1w
2BINZySbPsvUC/4FgGmNmTH6s+V7YYpCqyIwQ1cLrsxvMdwKAXYslaVMhxblRk3q
Xb5qWUv2Nb4dEMfBsWxqAPzbCuXAw2jlizyRSsqjotEcfxtYgjMOTCX41y1JpaUe
/tU+zKHBtCTwKvT05cFiXWAXbkyMDGg4yk79GumT5E1qC8f3eOMZhTcoiT599rMb
P+B3f9ka2W3g1jkAiOPIz5sQRO93VggamjBXIR/I5JC3ljJpu86uwJuWlTSJ1PE3
wZPxcNgb1Od60uc80ye1O1WTV0Wz+xYo2OnqAcBsoHti4f3o6A1SL7RCIUtdIflE
VQIDAQAB
-----END PUBLIC KEY-----

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
TLS 1.0 **TLS 1.1** **SSL (all versions)**	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. 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.
SHA-1	No new implementations are permitted, and all existing implementations must be upgraded to at least a SHA-2 compliant algorithm.
OAuth 1	Inland Revenues does not and will not support OAuth 1.
RSA	The RSA suite of signing algorithms are still supported but use of them is discouraged in favour of the ecliptic curve signing algorithms. Refer to section 2.4 for the specific algorithms.
Token Expiry	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
1.6	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"
1.7	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://
1.8	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.
1.8.1	23/01/2018	2.1.4	Commentary added for refresh token validity
1.9	05/02/2018	2.2	Further detail for Service Providers offering desktop solutions regarding OAuth for Native apps.
2.0	22/02/2018	New section 2.1.3.4	Insert new section describing the user of the consent screen
		New section 2.1.6	Insert new section describing the Validate Token Service
		Use of 'State' in transaction calls	Security recommendation based upon RFC 6749
		Section 2.2	RFC 8252 Section 7.3 IRD unable to allow any port for return uri
2.1	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.
2.2	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
2.3	28/05/2019	Multiple	Amend end point references to TEST4.services~
2.4	1/06/2019	P 14 Table	Minor alts to make http 401 error visible STATE parameter limit added.
2.5	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.
2.6	23/06/2020	StartLogon can be optional	Amend StartLogon value in table for Client Signed JWT to be optional
2.7	09/09/2020	Inland Revenue upgrade of OAuth system	Inland Revenue updated the backend XIAMS system for OAuth Examples updated with minor changes to order of fields returned. Section 2.1.7 – Refresh tokens do not expire.
2.8	23/12/2020	2.1.3-5	Update to OAuth2.0 API error messages. Includes a new HTTP status codes, error descriptions and advises clients on remedial action.