

**Application Program Interface (API) Authorization User Guide**  
for  
**eServices APIs**

**Internal Revenue Service (IRS) eServices**

# TABLE OF CONTENTS

---

<b>1.1 OVERVIEW OF KEY CONCEPTS .....</b>	<b>4</b>
1.1.1 Client ID .....	4
1.1.2 JWKs with use of “x5t” Parameter and “x5c” Parameter .....	4
1.1.2.1 Example of JWK Keys.....	4
1.1.3 Access Types – Authorization Flows .....	6
1.1.4 Consent.....	7
1.1.5 Access Token and Refresh Token .....	7
1.1.6 JWT BEARER Grant Type .....	7
1.1.7 AUTHORIZATION CODE Grant Type.....	8
1.1.8 Consumption limit.....	8
1.1.9 Understanding Authorization Flows Endpoints.....	8
<b>SECTION 2 - A2A AUTHORIZATION PROCESS .....</b>	<b>9</b>
2.1 A2A CLIENT ID REGISTRATION .....	9
2.2 Consent.....	9
2.3 Access Token Generation for A2A Access Flow .....	9
<b>SECTION 3 - ISP AUTHORIZATION PROCESS .....</b>	<b>14</b>
3.1 ISP - CLIENT ID REGISTRATION .....	14
3.2 Consent.....	14
3.3 Access Token Generation for ISP Flow .....	14
<b>SECTION 4 - ERROR RESPONSE REFERENCE .....</b>	<b>19</b>
4.1 Error Response Codes.....	19
<b>APPENDIX A - ABBREVIATIONS AND ACRONYMS .....</b>	<b>23</b>
<b>APPENDIX B - E-SERVICES API OVERVIEW .....</b>	<b>24</b>

## LIST OF TABLES

---

Table 1-1: Authorization Types .....	7
Table 2-1: Token Endpoint - Parameters.....	11
Table 2-2: Success Response Parameters .....	12
Table 4-1: Response Error Codes.....	19
Table A-1: Abbreviations and Acronyms .....	23
Table B-1: List of Available e-Services APIs .....	24

## LIST OF FIGURES

---

Figure 1-1: JWK Example .....	5
Figure 2-1: A2A OAuth Flow – Diagram .....	10
Figure 2-2: Login endpoint HTTPs request - Example.....	11
Figure 2-3: Access Token\Refresh Token POST request – JWT Grant Type Examples .....	11
Figure 2-4: Access Token Successful Response - Example.....	12

Figure 3-1: Consent Revoke Screen .....	14
Figure 3-2: ISP Client App OAuth Flow – Diagram.....	15
Figure 3-3: Token Request with a Code Grant Type- Example .....	16
Figure 3-4: Access Token/Refresh Token POST Requests – Example .....	16
Figure 3-5: Access Token Response - Example.....	17
Figure 3-6: Accessing API Resources.....	18
Figure 4-1: Error Body .....	19

## HIGH LEVEL OVERVIEW

---

This guide provides the instructions on how to register and get the proper authorization to utilize the assets of the three e-Services APIs: TIN Matching (TINM) API, Transcript Delivery System (TDS) API, and the Secure Object Repository (SOR) API. Its scope is limited to the registration and authorization process. This guide does not discuss the capabilities of the APIs. Details pertaining to each of the APIs and what capabilities they offer, are provided in their associated user guides. An Overview of each API is provided in Appendix B of this document. The authorization process is token based authentication scheme following the OAuth authorization access framework. All consumers of the e-Services APIs must be registered clients and have a Client ID. A Tax Professional consent is required to access the APIs on their behalf.

### 1.1 OVERVIEW OF KEY CONCEPTS

#### 1.1.1 Client ID

Client ID is the unique identifier of the e-Services API consumer application. The consumers of the e-Services API must be approved API clients and have a Client ID. Client ID will be provided to you at the time of registration. Registration details are described in Section 2.1 and Section 2.3 of this guide.

#### 1.1.2 JWKs with use of “x5t” Parameter and “x5c” Parameter

A JSON Web Key Set (**JWKs**) that represents a cryptographic key is used for e-Services API authentication. It contains a public key that validates the API consumer application. JWK will have the following criteria:

- **JWKs should contain a public key using RSA algorithm.** RSA provides a key ID for key matching purposes.
- Should contain X.509 certificate using both “x5t” (X.509 SHA-1 Thumbprint) and “x5c” (X.509 certificate Chain) parameters.
- **The minimum set of attributes expected in the JWK are:**
  - “kty”: Key Type
  - “kid”: Key ID
  - “use”: “sig” Public Key Use
  - “n”: the modulus
  - “e”: “AQAB” the public exponent
  - “x5c”: X. 509 Certificate Chain
  - and “x5t”: X.509 Certificate SHA-1 Thumbprint
- During registration you must provide the JWK Set file. **Note: if any of the above attributes is missing from the JWK, the JWK will be invalid.**
- **It is your responsibility to keep track of the JWK expiration date and provide a new one once the current JWK expires.**
- For more information on JWK visit [RFC 7517](#)

##### 1.1.2.1 Example of JWK Keys

The following figure 1-1 contains example of RSA key represented as JWKs.

Figure 1-1: JWK Example

```
{
  "keys": [
    {
      "kty": "RSA",
      "kid": "20190607",
      "use": "sig",
      "n": "ruZaAO0J6QGmpXOJNs9xxgMRXiU6DZTIQD2ZVt2fGaiTY_hu9HdmH3IBO6lc32rp2O201pzwCDsloOmPXr8-
gmI5oprV8a-pAcfbhNzi4wr6FpY3aLafjUmPDWjNuiGZxSwsiyb7OJdpGHipmC1Mz-
fh4ZZNbjP51dASWs5WOFlyMDYQ4W6mKIDQ8ku9J5rD_RDzUnjH3RRugG8q0moETXoFZgUJp3jEggbGOPoMzRqFSp9_gXvGchzhG8
fkzJzXCi1-fzGozjzCehlWw_h262xoWOKb0PbD58cUNehmVg6Y-
px24f3echmsTdt3Xfq7V1X7F94ytYf4I_kSvD1vwspYFxl6uBnikPTu66uI2GuLjdGrQ1FV94pFvEpWDh0NMChRZkzs1VW6tkZZ05jc15Yu
8E8RJBeoV2zsPDJ6omvDP_JdfK9MXVmHj0OA8GCBTLi5kXvSuDqN0CbUaO0xvPzZw4aFK8dqv1UL7MO184J-3ixwfGKr1-pkZMYC5",
      "e": "AQAB",
      "x5c": [
        "MIIDSDCCAhigAwIBAgIWAwsyWmY+MA0GCSqGSIb3DQEBCwUAMB4xHDAaBgNVBAMME2VzZXU2LWFwaS10ZXN0LXRvb2wwHhc
NMTkwNTAxMDQwMDAwWWhcNMjAwNTAxMDQwMDAwWjAUMRIwEAYDVQQDDAlzY290dC1yc2EwggGiMA0GCSqGSIb3DQEBAQ
UAA4IBjwAwggGKAoIBgQCu5loA7QnpAaalc4k2z3HGAXFeJToNIMhAPZlW3Z8ZqINj+G70d2YfcgE7qVzfaunY7bTwnPAIOyWg6Y9ev
z6AyXmimtXxr6k8x9uE3OljCvoWljdosB+NSY8NaM1SIZnFLCyLjvs4l2kYeKmYLUzP58fhlk1uM/nV0BJazlY4UvIwNhdhbqYqUNDyS70
nmsP9EPNSeMfdFG6AbyrSagS1egVmBQmneMSCBsY4+gzNGoVKn3+Be8ZyHOEbx+STMnNcjX5/MajOPMJ6GVbD+HbrbGhY6RvQ
9sPkFqX16GZWDpj6nHbh/d5yGaxN21rdd+rtXVfsX3jK1h/gj+RK8PW/CylgXEvq4GeKQ9O7rq4jYa4uN0atDUVX3ikW8SIYOHQ0wkFF
mTOzVVbq2RlnTmNzXli7wTxEkF6hXbOw8MiPqia8M/8l18r0xdWYePQ4DwYIG0uLmQrFK4Oo3QjtRo7TG8/NnDhoUrx2q/VQvsw7Xz
gn7eLHB8YqvX6mRlxgkLkCAwEAATANBgkqhkiG9w0BAQsFAAOCAQEAIrr3UEV2yB73S3XYECI48A6nqGjpu1ufzFwGmncchq/E8aVBm
vedAc4ONwy4XcNUJP0CeQtKGyOR0SPkWoCsBx+F+8hilan8fb4buOqfyWefw9w8crkmWvl6wTr0qOgRQeGgt8LeOYmTslf7UW86J
PIgy5KhPUE6EEdfO5ourekjXJo14BV5+8Xrr6wLZLd/t+qKiHHzLioCHqv4J0Pbut9UtUjy9OWsxDmsau/plhPsnMxOC8FsZxv8rrFFGhx2
5BG3k0VvSazPVsYzfjipN0ahPr+1z0L5tjIEXJ9qDF8fc0YGkhlbjFinya60ztI8aUirmN61mCwaXYGwWiyGvRdgrYtbq9Xj24Gev79pWjybH
v2Wchg4oYHw3Yd3wDxFljmAzNLTfHBSHjq7EdpSu8k52TqZsrHqp3dE5ehG4YnJ+J/81M5ItE9SA40365q1RPURWMtymdNLS1VSjAb
Hw2CRXbybpNqgZ9Q5zM1Aku+NMSoNEj5H+gmUV"
      ],
      "x5t": "7a02e132b81e408e52d83506ed5edd1a89c719a2"
    }
  ]
}
```

### 1.1.3 Access Types – Authorization Flows

The e-Services API offers two flows to obtain app authorization: **Intermediate Service Provider (ISP) process flow** and **Application-to-Application (A2A) process flow**. Both flows implement the OAuth 2.0 open authorization protocol.

**Intermediate Service Provider** and **A2A** will enable the software to obtain access on behalf of tax professionals by orchestrating an approval interaction with e-Services API using access token in lieu of username and password, or by allowing the software to obtain access on its own behalf. The access token is a string denoting a specific scope, lifetime, and other access attributes. The access token is authorized to perform wanted actions such as retrieving SOR messages, performing a TIN match, requesting a TDS transcript, etc.

- **ISP Authorization** – ISP (Intermediate Service Provider) authorization happens when a TaxPro is using a 3<sup>rd</sup> party software application for his/her work. These software products are unknown to eservices and may have features that retrieve data from the IRS via the eServices APIs(s) on behalf of the user. This is done while the user has an active session with the 3<sup>rd</sup> party software, meaning the user is at the keyboard while this is happening. With ISP, the organization is selling their product to others.
- **A2A Authorization** – A2A authorization is for eServices users to give authority to their organization's A2A process to run API transactions on his/her behalf. A2A authority must be setup ahead of time of the API(s) consumption, since the A2A process are headless in nature, meaning the user is not involved or present at the time of execution. With A2A, the organization is not selling their product to others.

The **key technical variations** between **ISP** process flow and **A2A** process flow are:

1. ISP requires Client ID and JWT Client Credentials (signed with private key) to generate the access token. In addition, it will require a redirect Uniform Resource Identifier (URI). With this flow, a tax professional's consent should be in place to approve access of protected resources by the client application. Therefore, ISP involves authorization from the tax professional. The tax professional (the user) who owns the TINM, TDS, or SOR resources; such as taxpayer's tin, name, taxpayer transcript, etc..., must login to provide authorization and consent. An authorization code is generated and automatically redirect to the client application using URL provided during registration. In exchange, the IRS authorization server will provide the ISP the access token. The ISP uses the generated access token to make API calls.
2. A2A process requires Client ID, JWKS, and two JWT Bearer tokens (signed with private key); one for Client Credentials and one for User Credentials to generate the access token. The organization must be registered in e-Services. A tax professional's consent should already be obtained prior to using the A2A flow. This is suitable for machine-to-machine interaction and does not require a tax professional to log in. A2A application uses the generated access token to make API calls. For more information about e-Services applications, visit the e-Services site at <https://www.irs.gov/e-services>

Table 1-1 provides the main differences between the two types of authorization flows.

Table 1-1: Authorization Types

Authorization Flows								
FLOW	MUST HAVE ESAM APP	REQUIRES TAX PRO CONSENT	REQUIRES ACCESS TOKEN	REQUIRES JWKS WITH USE OF CERTIFICATE	REQUIRES CLIENT ID	REQUIRES JWT BEARER TOKEN	REQUIRES AUTHORIZATION CODE	PROVIDES REFRESH TOKEN
ISP Auth	Yes	Yes – during the ISP Flow	Yes	Yes	Yes	Yes: Only Client Credentials JWT type	Yes	Yes
A2A Auth	Yes	Yes – Prior to accessing A2A Flow	Yes	Yes	Yes	Yes: Client JWT and User JWT	No	Yes

### 1.1.4 Consent

The consumers of the e-Services API are accessing TINM, TDS, and SOR resources on behalf of tax professionals. Therefore, a tax professional consent must be obtained. When an application, either ISP or A2A, needs to get a new access token from IRS authorization server, the application is required to get user’s authorization and consent.

With the Intermediate Service Provider process, the tax professional logs in and grants consent to the ISP application’s Client ID to access resources. With A2A process, a consent for A2A application’s Client ID should be obtained prior to making requests through the e-Services API. In the case where a user consent is not available for A2A process, the request will be rejected and the A2A application should have the user provide consent through the Consent App.

### 1.1.5 Access Token and Refresh Token

The IRS authorization server issues the access tokens to each organization A2A or ISP as credentials to access the e-Services API resources. This access token is a string representing an authorization in an opaque format. The access tokens will have specific scopes and durations of 15 minutes lifespan, with a refresh token. Refresh tokens are used to obtain a new access token when the current access token becomes invalid or expires. Refresh token is also issued by the IRS authorization server.

### 1.1.6 JWT BEARER Grant Type

For A2A process flow, the “JWT Grant Type” is used as a means for requesting an OAuth 2.0 access token and refresh token. The A2A app must issue two JWTs (one for Client, and another one for User) and exchange them for an access token with the IRS authorization server. The JWT must conform to JSON Web Signature JWS with the following claims:

- **iss** (issuer) – Identifies who issues the token. Must include the Client ID obtained at registration.
- **sub** (subject) – Subject of the token. Must include:
  - The **Client ID** for **Client JWT** token type
  - The **User ID** for **User JWT** token type
- **aud** (audience) – the IRS authorization server. The token endpoint of the auth server (See Section 1.1.8)
- **iat** (issued at time) – *Optional*. Issued at time. Numeric value of the time the token was created
- **exp** (expiration time) – Numeric value of the time when the token expires. It must be valid for 15 minutes.

- **jti** (JWT ID) – *Required*. Provides unique identifier for the JWT. It prevents the JWT from being replayed. This is required by the IRS API Gateway.

**Note:** In addition to the required claims above, the JWT header should include the “alg” and the “kid” claims, otherwise the JWT will be invalid.

The JWT Grant type request will have the following parameters:

- grant\_type – required, value should be “jwt-bearer”
- Assertion – required, JWT value
- Client assertion type – required, value should also be “jwt-bearer”
- Client assertion – required, JWT value

### 1.1.7 AUTHORIZATION CODE Grant Type

For ISP consumer type, the “Authorization Code Grant Type” is used as a means for requesting an OAuth 2.0 access token and refresh token. The authorization code request will have the following parameters:

- grant\_type – required, value should be “authorization\_code”
- code – required, the authorization code value received from the Authorization server
- Client assertion type – required, the JWT value
- Client assertion – required, the JWT value
- state – optional but allowed as a pass param back to the client app

**In addition to the parameters above**, the ISP app must issue **one Client Credentials JWT**. ISP will provide this JWT along with the Auth Code, to exchange them for an access token/refresh token with the IRS authorization server. The Client Credentials JWT should have the claims described in [Section 1.1.6](#) where **sub** claim will have the **Client ID** only.

### 1.1.8 Consumption limit

There is a limit on the number of eServices API calls a consumer can make. If the number of requests exceeds that limit, you will receive an error and a blackout period of 10 minutes duration will be imposed. The error message description you will receive is : *“Number of permitted requests has been exceeded. A 10-minute blackout is now in effect”*, with a Status code 429.

### 1.1.9 Understanding Authorization Flows Endpoints

The authorization flows endpoints are the URLs you use to make OAuth authentication requests to IRS e-Services APIs.

It is imperative that you use the proper e-Services OAuth endpoint when issuing authentication requests in your application.

The primary endpoints for **ISP** flow are:

- For authorization: <https://api.www4.irs.gov/auth/oauth/v2/authorize>
- For token requests: <https://api.www4.irs.gov/auth/oauth/v2/token>

The primary **A2A** endpoint is:

- For token requests: <https://api.www4.irs.gov/auth/oauth/v2/token>

All endpoints require secure HTTP (HTTPS). Each OAuth flow defines which endpoints you need to use and what request data you need to provide.

**To verify authentication in the test environment URL**, use “<https://api.alt.www4.irs.gov>” instead of “<https://api.www4.irs.gov>” in all the OAuth endpoints listed above.



## SECTION 2 - A2A AUTHORIZATION PROCESS

---

### 2.1 A2A CLIENT ID REGISTRATION

To become an authorized e-Services API consumer, you must be approved to receive a Client ID and follow e-Services A2A process flow. It is imperative to know the specifications of ISP App vs. A2A authorizations as described in [Section 1.1.3](#).

To receive a Client ID for the A2A authorization flow, the following actions must take place:

1. The principal and delegated users must complete Secure Access registration for e-Services. As a reminder, tax professional's consent should be obtained prior to using the A2A flow. For additional information, see Section 2.2 in this user guide.
2. Once the registration process is completed, the principal will need to complete an API Client ID Application. While completing the application, you will need to provide the JWKs file with use of valid X.509 digital security certificate as described in [Section 1.1.2](#). The certificate will be validated during the application process.
3. Once the application is completed, the principal or delegated user will need to sign in and obtain the Client ID(s) issued to the firm/organization. The Client ID(s) will be listed on the Application Details or Application Summary pages.

For any questions related to the API Client ID Application, contact the e-Help Desk at 1-866-255-0654.

### 2.2 CONSENT

For A2A client IDs to run e-Services API transactions on behalf of e-Services users, those Tax Professionals must first grant access to A2A client id before a client can request an access token on their behalf. Tax Pro must perform the following two steps to grant access:

1. Tax Professional login to IRS Consent App.  
<https://la.www4.irs.gov/esrv/consent/>
2. Tax Professional sets up A2A access.

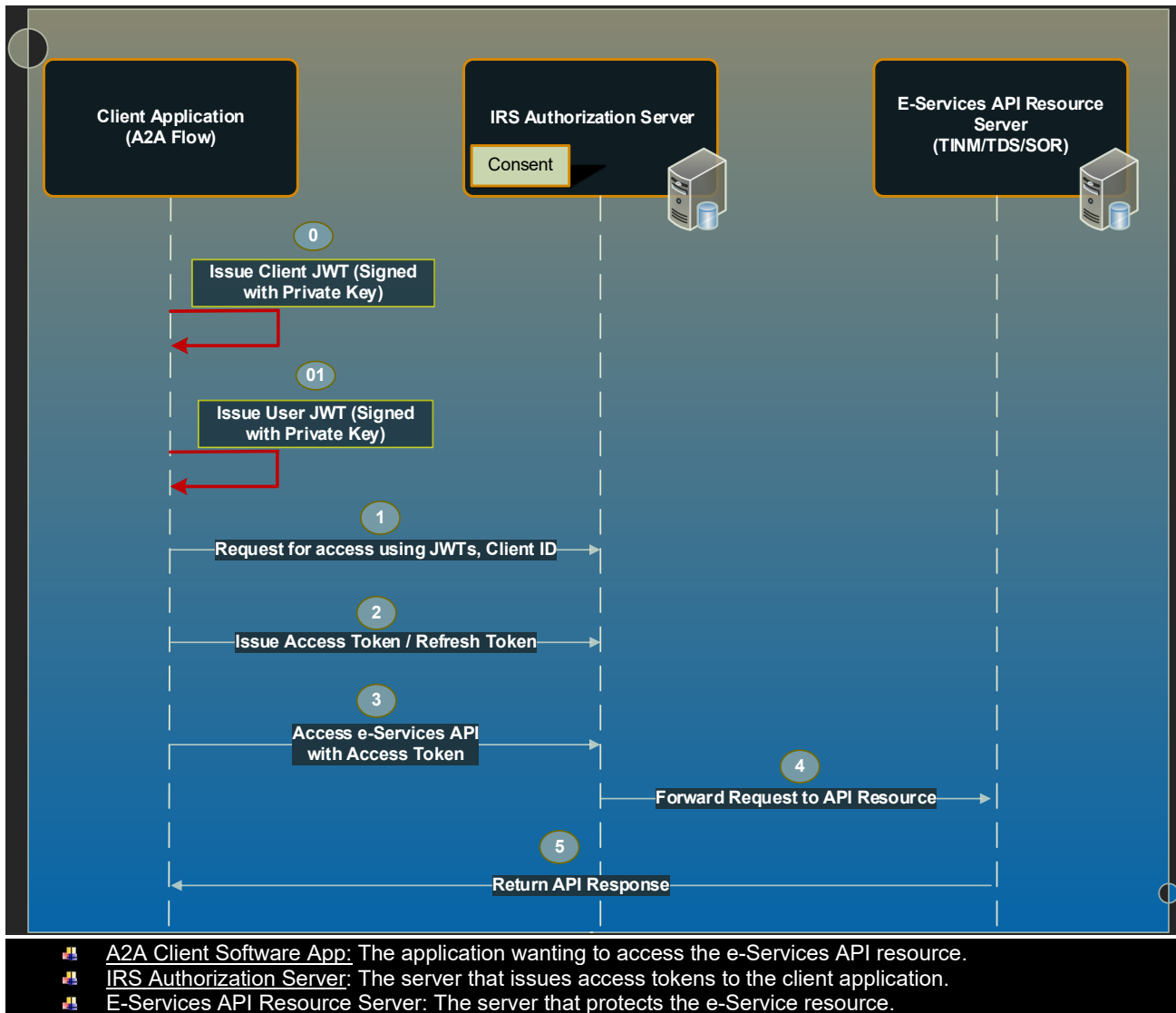
A2A login endpoint will check if consent exists:

1. If consent exist, A2A login endpoint will provide access token contains consent is true.
2. If consent does not exist, A2A login endpoint returns 401 Unauthorized response with reason for denial.

### 2.3 ACCESS TOKEN GENERATION FOR A2A ACCESS FLOW

In A2A flow, the client application requests authorization from the IRS server for API access. The IRS server verifies user's consent and requires the application to provide a Client ID, and two JWTs. The application uses Client ID and signed (with private keys) JWT Bearer tokens to request an access token. Figure 2-1 shows steps including the generation of the access token.

Figure 2-1: A2A OAuth Flow – Diagram



The A2A flow illustrated in figure 2-3 consists of the following steps including the generation of the access token:

**Step 0 and 01** - The A2A Client App must issue two JWTs and they must be signed with private keys to validate assertion. The JWTs should be in JWT token format using the claims stated in [Section 1.1.6](#)

**Step 1** - The A2A Client App requests API access using the **URL token endpoint**:

<https://api.www4.irs.gov/auth/oauth/v2/token>

**Step 2** - If authenticated successfully, the authentication server responds to the app and exchange Client ID & validated JWKS certificate (received during registration process) & validated signed JWTs with an access token and refresh token. The access token is packaged into query parameter in a response redirect to the request.

The A2A Client App is required to provide the parameters in **Table 2-1** in the authorization HTTP header.

**Table 2-1: Token Endpoint - Parameters**

PARAMETER	DESCRIPTION
grant_type	Required: Value must be set to “urn:ietf:params:oauth:grant-type:jwt-bearer”
assertion:	Required: The assertion used as authorization grant. Must contain a single jwt. {app-signed-jwt}
client_assertion_type:	Required: The value is urn:ietf:params:oauth:client-assertion-type:jwt-bearer
Client_assertion:	Required: contains a single JWT. Must not contain more than one JWT

The example in Figure 2-2 shows A2A authorization URL **login endpoint**.

**Figure 2-2: Login endpoint HTTPs request - Example**

```
POST /token.oauth2 HTTP/1.1
Host: api.irs.gov
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer
&assertion=eyJhbGciOiJIJFZlIiwiaWF0IjE2In0.
&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer
&client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjE2In0
```

The client app sends over an access token request using a client ID and signed JWT tokens. Example in Figure 2-3 shows an access token /refresh token POST requests send the JWT tokens. **Note:** The example below is using the test endpoint. JWT is represented as `XXXX.XXXX.XXXX` be sure to replace them with your JWTs before running the example.

**Figure 2-3: Access Token\Refresh Token POST request – JWT Grant Type Examples**

```
curl -k -POST https://api.alt.www4.irs.gov/auth/oauth/v2/token \
-H "Content-Type: application/x-www-form-urlencoded" \
-d @- <<EOF
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer
&assertion=XXXX.XXXX.XXXX
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=XXXX.XXXX.XXXX
EOF
```

Note `$refresh_token` needs to be replaced with a refresh token you received

```
curl -k -POST https://api.alt.www4.irs.gov/auth/oauth/v2/token \  
-H "Content-Type: application/x-www-form-urlencoded" \  
-d @- <<EOF  
grant_type=refresh_token  
&refresh_token=$refresh_token  
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer  
&client_assertion=XXXX.XXXX.XXXX  
EOF
```

The HTTP response with 200 (OK) status will contain the following parameters as defined in Table 2-2.

Table 2-2: Success Response Parameters

PARAMETER	DESCRIPTION
access_token	Access token that acts as a session ID that the application uses for making requests.
token_type	Value is Bearer for all responses that include an access token
refresh_token	Secret value. Used to obtain new access tokens. (3)
expires_in	The lifetime in seconds of the access token. For example, the value "900" denotes that the access token will expire in 15 minutes from the time the response was generated.

The basic structure of a response is a JSON object that holds the response information. Figure 2-4 shows an example of a successful response.

Figure 2-4: Access Token Successful Response - Example

```
HTTP/1.1 200 OK  
Content-Type: application/json; charset=UTF-8  
Cache-Control: no-store  
Pragma: no-cache  
  
{  
  "access_token": "<Access-Token> ",  
  "token_type": "Bearer",  
  "refresh_token": "<Refresh-Token> ",  
  "expires_in": 900  
}
```

**Step 3** - The redirect points the app's request back to the e-Services API resource server. The access token gets added to the header of the API request with the word *Bearer* followed by the token string.

**Step 4** - The e-Services API server checks the access token in the app's request and decides whether to authenticate the app.

**Step 5** - The e-Services API resource sends response successfully.

**Note:** Access token will expire after 15 minutes and a refresh token gets revoked after one hour. When revoke occurs, application will be redirected to authorization server.

## SECTION 3 - ISP AUTHORIZATION PROCESS

### 3.1 ISP - CLIENT ID REGISTRATION

To become an authorized e-Services API consumer, you will register your application to receive a Client ID and follow e-Services ISP Application flow. It is imperative to know the specifications of ISP vs. A2A authorizations as described in [Section 1.1.3](#).

To receive a Client ID for the ISP App authorization flow, the following actions must take place:

1. The principal and delegated users must complete Secure Access registration for e-Services.
2. Once the registration process is completed, the principal will need to complete an API Client ID Application. While completing the application, you will need to provide the JWKs file with use of valid X.509 digital security certificate as described in [Section 1.1.2](#). The certificate will be validated during the application process. The system will also require you to provide a redirect URL.
3. Once the application is completed, the principal or delegated user will need to sign in and obtain the Client ID(s) issued to the firm/organization. The Client ID(s) will be listed on the Application Details or Application Summary pages.

For any questions related to the API Client ID Application, contact the e-Help Desk at 1-866-255-0654.

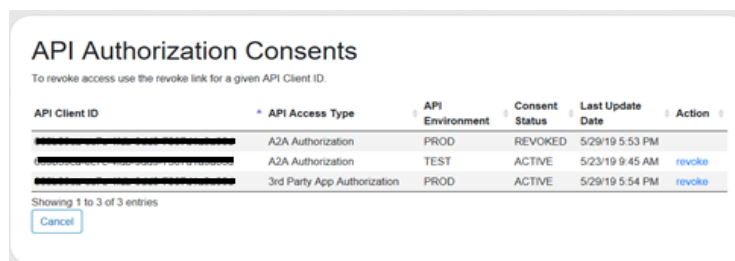
### 3.2 CONSENT

When the resource owner (Tax Professional) reaches the API entry point, she or he will be prompted for consent to authorize or deny the application's access on their behalf, **via the** user web browser. The following steps must occur:

1. Tax Professional navigates to IRS Consent App.
2. Tax Professional sets up ISP access. This will create a user grant which represents the user's consent.

**Note** that once consent has been given, the user won't be redirected to the consent app during subsequent logins until consent is revoked explicitly. See Figure 3-1

Figure 3-1: Consent Revoke Screen



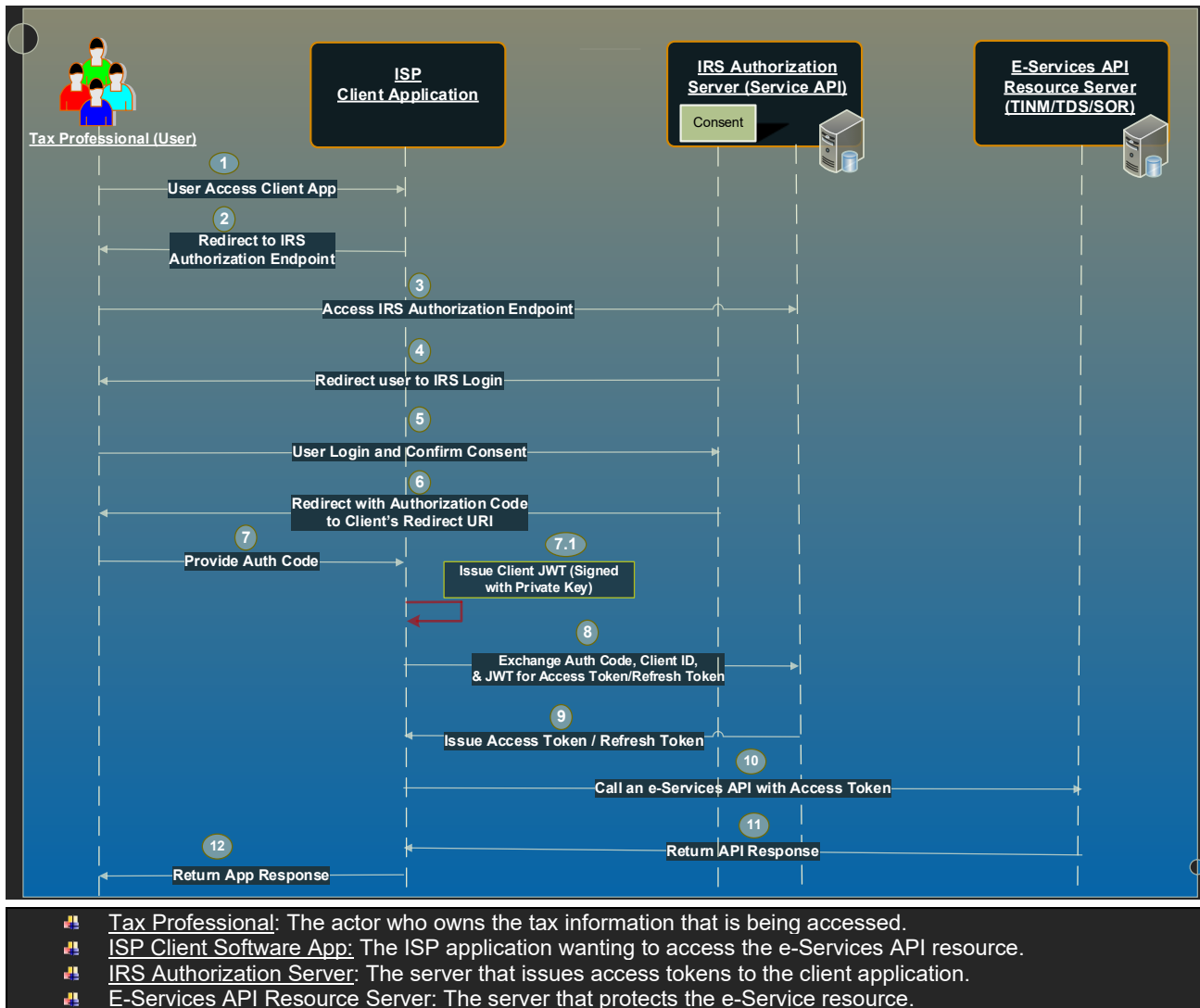
The screenshot shows a web interface titled "API Authorization Consents". Below the title is a subtitle: "To revoke access use the revoke link for a given API Client ID." The main content is a table with the following columns: "API Client ID", "API Access Type", "API Environment", "Consent Status", "Last Update Date", and "Action". There are three rows of data. The first row has a redacted API Client ID, "A2A Authorization", "PROD", "REVOKED", and "5/29/19 5:53 PM". The second row has a redacted API Client ID, "A2A Authorization", "TEST", "ACTIVE", "5/23/19 9:45 AM", and a blue "revoke" link. The third row has a redacted API Client ID, "3rd Party App Authorization", "PROD", "ACTIVE", "5/29/19 5:54 PM", and a blue "revoke" link. Below the table, it says "Showing 1 to 3 of 3 entries" and there is a "Cancel" button.

API Client ID	API Access Type	API Environment	Consent Status	Last Update Date	Action
[REDACTED]	A2A Authorization	PROD	REVOKED	5/29/19 5:53 PM	
[REDACTED]	A2A Authorization	TEST	ACTIVE	5/23/19 9:45 AM	<a href="#">revoke</a>
[REDACTED]	3rd Party App Authorization	PROD	ACTIVE	5/29/19 5:54 PM	<a href="#">revoke</a>

### 3.3 ACCESS TOKEN GENERATION FOR ISP FLOW

In ISP flow, the client application requests authorization from the IRS server for API access. The IRS server verifies user's consent and sends the application an authorization code. The application uses the authorization code to request an access token and a refresh token. Figure 2-3 shows the steps including the generation of the access token.

Figure 3-2: ISP Client App OAuth Flow – Diagram



The ISP flow illustrated in Figure 3-2 consists of the following steps including the generation of the access token:

**Step 1** - The user accesses the Intermediate Service Provide ISP App.

**Step 2** – The ISP App redirects the user to the **URL** authorization **endpoint**:

- [https://api.www4.irs.gov/auth/oauth/v2/authorize?client\\_id=\[CLIENT ID\]&response\\_type=code](https://api.www4.irs.gov/auth/oauth/v2/authorize?client_id=[CLIENT_ID]&response_type=code)

**Step 3** - The user accesses the authorization endpoint link above.

**Step 4** – The Authentication server redirects the user to provide consent. User will be prompted to authorize the application through the Consent App.

**Step 5** - User authorizes the application and provide consent.

**Step 6** – User redirected with authorization code using redirect URI.

**Step 7** – Through the redirect URI, user provides the authorization code to the ISP App. **Note:** *The auth code has a lifespan of 10 minutes*

Figure 3-3: Token Request with a Code Grant Type- Example

```
POST /token HTTP/1.1
Host: api.irs.gov
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=Splx10BeZQQYbYS6WxSbIA
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
client-assertion-type%3Ajwt-bearer&
client_assertion=eyJhbGciOiJIUzI1NiIsImtpZCI6IjIyIn0&
```

**Step 8** – The ISP App receives the authorization code and exchanges the auth code, the Client ID & JWT Bearer Token (For Client Authentication) for Access Token. Figure 3-4 shows an access token/refresh token requests using curl.

The token endpoint will include:

- API **token** endpoint: <https://api.www4.irs.gov/auth/oauth/v2/token>
- `grant_type = code`: Value must be set to “authorization\_code”
- `code = $auth_code`: the code received from the Authorization server
- `Client_asseration_type=` urn:ietf:params:oauth:client-assertion-type:jwt-bearer
- `Client_asseration=` your JWT value

Figure 3-4: Access Token/Refresh Token POST Requests – Example

*Note: JWT(s) will be represented as XXXX.XXXX.XXXX be sure to replace them with your JWTs before running the examples*

```
curl -k -POST https://api.alt.www4.irs.gov/auth/oauth/v2/token \
-H "Content-Type: application/x-www-form-urlencoded" \
-d @- <<EOF

grant_type=authorization_code
&code=$auth_code
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=XXXX.XXXX.XXXX

EOF
```

*Note \$refresh\_token needs to be replaced with a refresh token you received*



```
curl -k -POST https://api.alt.www4.irs.gov/auth/oauth/v2/token \  
-H "Content-Type: application/x-www-form-urlencoded" \  
-d @- <<EOF  
grant_type=refresh_token  
&refresh_token=$refresh_token  
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer  
&client_assertion=XXXX.XXXX.XXXX  
EOF
```

**Step 9** - If authenticated, the authentication server responds to the app and exchange Auth code, Client ID & JWT with an access token. The access token is packaged into query parameter in a response redirect to the request. Figure 3-5 shows an example of access token response.

Figure 3-5: Access Token Response - Example

```
HTTP/1.1 200 OK  
Status: 200 OK  
Content-Type: application/json; charset=utf-8  
  
{  
  "access_token" : "<Access_Token>"  
  "token_type" : Bearer  
  "refresh_token" : "<Refresh_Token>,"  
  "expires_in": 900 ,  
}
```

**Step 10** - The ISP App sends over an API request using access token. The redirect points the app's request back to the e-Services API resource server. The access token gets added to the header of the API request with the word *Bearer* followed by the token string.

Figure 3-6: Accessing API Resources

```
curl -v -X GET https://api.www4.irs.gov/esrv/api/sor/messages \
-H "Accept: application/json" \
-H "Authorization: Bearer $ACCESS_TOKEN"
```

```
curl -k -X POST https://api.alt.irs.gov/esrv/api/tinm/request \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $access_token" \
-d '{"tinType":"UNKNOWN", "tin":"123456780", "name":"Example Taxpayer}'
```

**Step 11** - The e-Services API server checks the access token in the app's request and decides whether to authenticate the app. If **successful response with 200** (OK), the e-Services API resources sends the response successfully to the ISP App.

**Step 12** – The ISP App sends response successfully to the user.

**Note:** access token will expire after 15 minutes and refresh token gets revoked after one hour. When revokes occur, the ISP needs to get new access token and refresh token by taking the user back to step 2.

## SECTION 4 - ERROR RESPONSE REFERENCE

### 4.1 ERROR RESPONSE CODES

In both authorization flows; A2A and ISP App, if the access token request is invalid, such as redirect URL did not match the one used during authorization or the Client Id does not match the one at registration, then the server needs to return an error response.

When errors occur, the authorization server responds with HTTP error code with *error* and *error\_description* parameters. The basic structure of an error response body is a JSON object. See example in Figure 4-1.

Figure 4-1: Error Body

```
{
  "error code": "ESRV306",
  "error_msg": {
    "error": "invalid client",
    "error_description": "The given JWT for client authentication is invalid."
  }
}
```

Table 4-1 shows summary of the error codes with their corresponding description and HTTP status codes.

Table 4-1: Response Error Codes

RESPONSE ERROR CODES			
HTTP STATUS CODE	ERROR CODE DESCRIPTION	MEANING	LOCATION
400 Bad Request	invalid_request	The request is missing a required parameter, includes unsupported parameter value, repeats a parameter, includes multiple credentials, utilizes more than one mechanism for authenticating the client.  Invalid Auth Code No Redirect URI Invalid refresh token, expired, revoke.	
	<b>Error Code</b>	<b>Error Message</b>	
	103	{ "error": "invalid_request", "error_description": "Missing or duplicate parameters" }	Authorize, Token
	110	{ "error": "invalid_request", "error_description": "The session has expired or already been granted. The login process has to be repeated to be successful" }	All

	112	{ "error": "invalid_request", "error_description": "the code_challenge or code_challenge_method is invalid" }	Register, Token, Authorize
	113	{ "error": "invalid_grant", "error_description": "The given grant is invalid" }	Token
	114	{ "error": "invalid_redirect_uri", "error_description": "One or more redirect_uri values are invalid" }	
	115	{ "error": "invalid_scope", "error_description": "No registered scope value for this client has been requested" }	Token
	116	{ "error": "unsupported_response_type", "error_description": "None of the supported response_types were used" }	Token
	119	{ "error": "unsupported_grant_type", "error_description": "The given grant_type is not supported" }	Token
	120	{ "error": "invalid_request", "error_description": "The id_token has missing claims or has expired" }	API
	121	{ "error": "invalid_request", "error_description": "The given JWT is invalid" }	Token
	124	{ "error": "access_denied", "error_description": "The resource_owner denied access to resources" }	Authorize
	132	{ "error": "invalid_request", "error_description": "The server configuration is invalid. Contact the administrator" }	All
	300	{ "error": "invalid_request", "error_description": "Client(_id) value could not be persisted" }	Register
	702	{ "error": "invalid_request", "error_description": "Client does not have access to this endpoint" }	Client Registration
	713	{ "error": "Refresh grant failed", "error_description": "Error in refresh grant - check rtoken expiry" }	Token
	715	{ "error": "token error", "error_description": "Refresh token Update Error" }	Token
	898	{ "error": "invalid_request", "error_description": "Service Not Found - Unsupported operation" }	All
	unauthorized_client	The client is not authorized to use this authorization type. Invalid JWT, invalid client id. Invalid callback URL.	

401 Unauthorized	<b>Error Code</b>	<b>Error Message</b>	
	201	{ "error": "invalid_client", "error_description": "The given client credentials were not valid" }	Authorize
	202	{ "error": "login_required", "error_description": "The resource owner could not be authenticated due to missing or invalid credentials" }	Authorize, Token
	205	{ "error": "invalid_request", "error_description": "The client certificate is not valid" }	
	306	{ "error": "invalid_client", "error_description": "The given JWT for client authentication is invalid." }	Token
	705	{ "error": "invalid_request", "error_description": "Null response returned from API resource server" }	API
	707	{ "error": "invalid_request", "error_description": "Bearer token has not expired" }	Token
	709	{ "error": "invalid_request", "error_description": "LDAP Error" }	Token
	717	{ "error": "assertion_error", "error_description": "Signature failed on validation" }	Token
	725	{ "error": "invalid_scope", "error_description": "Insufficient key scope" }	API
	990	{ "error": "invalid_request", "error_description": "Validation error" } [Note: Occurs when token Expired]	API
	991	{ "error": "invalid_request", "error_description": "Validation error" } [Note: Occurs on SCOPE Error]	API
	992	{ "error": "invalid_request", "error_description": "Validation error" } [Note: Occurs for missing or multiple token]	API
993	{ "error": "invalid_request", "error_description": "Validation error" } [Note: Occurs when token is disabled]	API	
403 Forbidden	203	{ "error": "invalid_request", "error_description": "SSL is required" }	All
	204	{ "error": "invalid_request", "error_description": "SSL with client authentication is required" }	All
429 Too Many Requests	Rate limit has been reached	The client app has reached the permissible API calls limit. Sent too many requests within the time limit than it is allowed to.	
	<b>Error Code</b>	<b>Error Message</b>	
	111	{ "error": "invalid_request", "error_description": "Number of permitted requests has been exceeded. A 10-minute blackout is now in effect" }	All
500 Not Found	Service error	Indicates a Service error. Authorization code expired, expired access token, expired refresh token,	
	<b>Error Code</b>	<b>Error Message</b>	

	000	{ "error": "invalid_request", "error_description": "The request failed due to some unknown reason" }	All
	701	{ "error": "invalid_request", "error_description": "Missing or invalid UID" }	API
	703	{ "error": "invalid_request", "error_description": "Missing or invalid eservices role" }	API
	711	{ "error": "invalid_request", "error_description": "Consent Error - Access Denied" }	Token
	719	{ "error": "invalid_request", "error_description": "Missing or invalid customer Id" }	API
	899	{ "error": "internal error", "error_description": "Consent Redirect Error" }	Authorize

## APPENDIX A - ABBREVIATIONS AND ACRONYMS

---

Table A-1 lists the abbreviations and acronyms used in this document.

Table A-1: Abbreviations and Acronyms

ABBREVIATION / ACRONYM	DEFINITION
API	Application Program Interface
A2A	Application to Application
CA	Certification Authority
CAF	Centralized Authorization File
EIN	Employer Identification Number
IRS	Internal Revenue Service
ISP	Intermediate Service Provider
OAuth	Open Authorization
PMO	Program Management Office
RAF	Reporting Agent File
REST	Representational State Transfer
SOR	Secure Object Repository
SSN	Social Security Number
TDS	Transcript Delivery System
TIN	Taxpayer Identification Number

## APPENDIX B - E-SERVICES API OVERVIEW

Table B-1 provides an overview of the e-Services Three APIs: TDS, TINM, and SOR.

**Table B-1: List of Available e-Services APIs**

API	DESCRIPTION
<b>TDS</b>	<p>The Transcript Delivery System (TDS) API allows authorized e-Services users to request return and account information electronically on behalf of their taxpayer and designees. TDS provides immediate return, account, record of account, wage and income, and verification of non-filing information in a masked form to taxpayers and their representatives.</p> <p>The TDS API provides the capability to retrieve a taxpayer's transcript using either the Centralized Authorization File (CAF) authority or the Reporting Agent File (RAF) authority. Both types of requests are described in Section 2, Operations.</p>
<b>TINM</b>	<p>The Taxpayer Identification Number (TIN) Matching Application Program Interface (API) allows a payer to submit a TIN, Name, TIN Type combination to be matched against IRS records. With the TIN Matching API, users can accomplish this by submitting a REST request in accordance with the technical guidance and data specifications provided in this document, which solicits an instant response. You may match a taxpayer's name with a Social Security Number (SSN), or a firm or business name with an Employer Identification Number (EIN). If you're not sure which identification number you have in your records, utilize Unknown and the system will attempt to match the name to both the EIN and SSN databases.</p>
<b>SOR</b>	<p>Secure Object Repository (SOR) API is an application designed to support requests for sensitive tax-related information. It provides a method to return sensitive, tax-related information that cannot be sent using ordinary e-mail to Authorized Registered users. The Taxpayer Identification Number (TIN) Matching system and Transcript Delivery System (TDS) can deliver documents to the SOR of a specific Authorized Registered user. The SOR stores documents for viewing and downloading by the Authorized Registered User.</p>