iRODS HTTP API v0.5.0

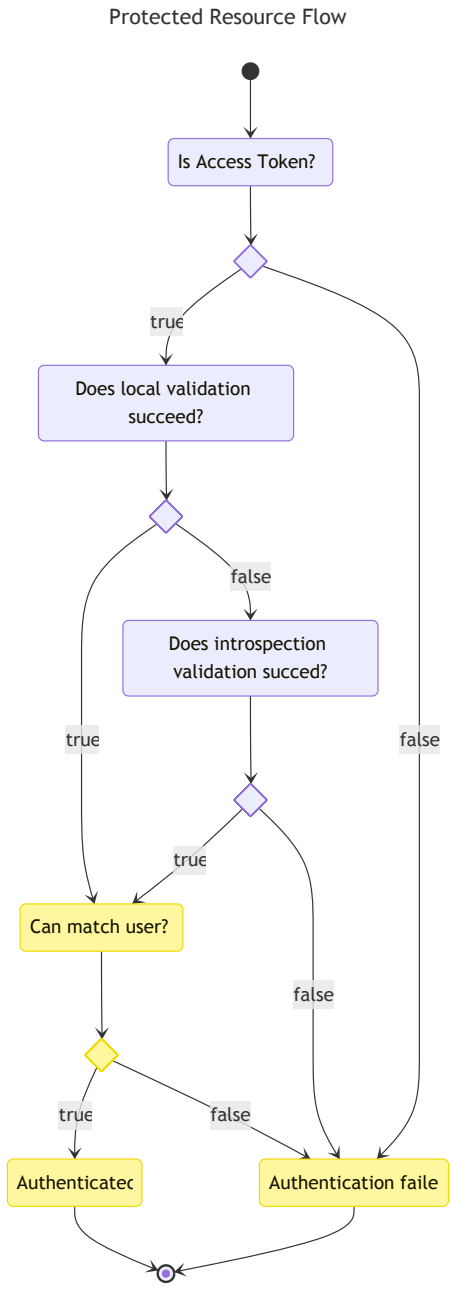
Kory Draughn, Chief Technologist
Martin Flores, Software Developer
iRODS Consortium

June 17-20, 2025
iRODS User Group Meeting 2025
Durham, NC

User Mapping - Overview

- Abstracted Mapping of JWTs to iRODS users
- Previously built-in mapping now plugins
  - user_attribute_mapping -> libirods_http_api_plugin-local_file.so
  - irods_user_claim -> libirods_http_api_plugin-user_claim.so
- Flexible mapping
  - e.g. query an external service for the mappings

User Mapping - Overview



User Mapping - Plugin Interface

- Simple C interface
- Full documentation in *interface.h*

```
1  /// Initializes the user mapping plugin.
2  int user_mapper_init(const char* _args);
3
4  /// Matches the given information to a user.
5  int user_mapper_match(const char* _param, char** _match);
6
7  /// Frees a C-string generated from the user mapping plugin.
8  void user_mapper_free(char* _data);
9
10 /// Executes clean-up for the user mapping plugin.
11 int user_mapper_close();
```

User Mapping - Local File Configuration Example

Old Configuration

```
...
"user_attribute_mapping": {
  "rodsBob": {
    "email": "bob@bobtopia.example",
    "sub": "a.very.real.sub",
    "phone_number": "56709"
  },
  "rodsAlice": {
    "email": "al-1s@wonderland.example",
    "sub": "a.different.sub"
  }
}
...
```

New Configuration

```
...
// Defines relevant information related to the User Mapping plugin system.
// Allows for the selection and configuration of the plugin.
"user_mapping": {
  // The full path to the desired plugin to load.
  "plugin_path": "/some/path/libirods_http_api_plugin-local_file.so",

  // The configuration information required by
  // the selected plugin to execute properly.
  "configuration": {
    "file_path": "/some-file.json"
  }
}
...
```

User Mapping - Local File Configuration Example

```
{
    "rodsAlice": {
        "email": "alice@example.org",
        "sub": "123-abc-456-xyz"
    },
    "rodsBob": {
        "email": "bob@example.org",
        "phone": "56709"
    }
}
```

Local File Mapping Example

User Mapping User Claim Configuration Example

Old Configuration

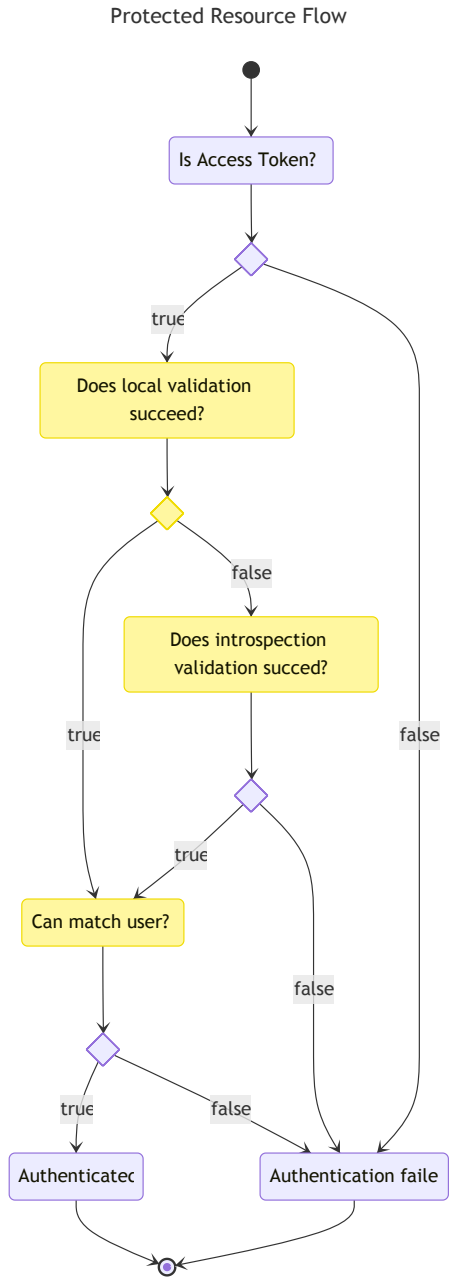
```
...
"irods_user_claim": "irods_username"
...
```

New Configuration

```
...
"user_mapping": {
  "plugin_path": "/some/path/libirods_http_api_plugin-user_claim.so",

  "configuration": {
    "irods_user_claim": "irods_username"
  }
}
...
```

Local JWT Access Token Validation



Local JWT Access Token Validation

- Validate Tokens locally using JWKs
- Similar to access token introspection
- Retrieve JWKs on first validation attempt
  - Minimal communication to OpenID Provider
- Support for more OpenID Providers

Local JWT Access Token Validation - Supported Algorithms

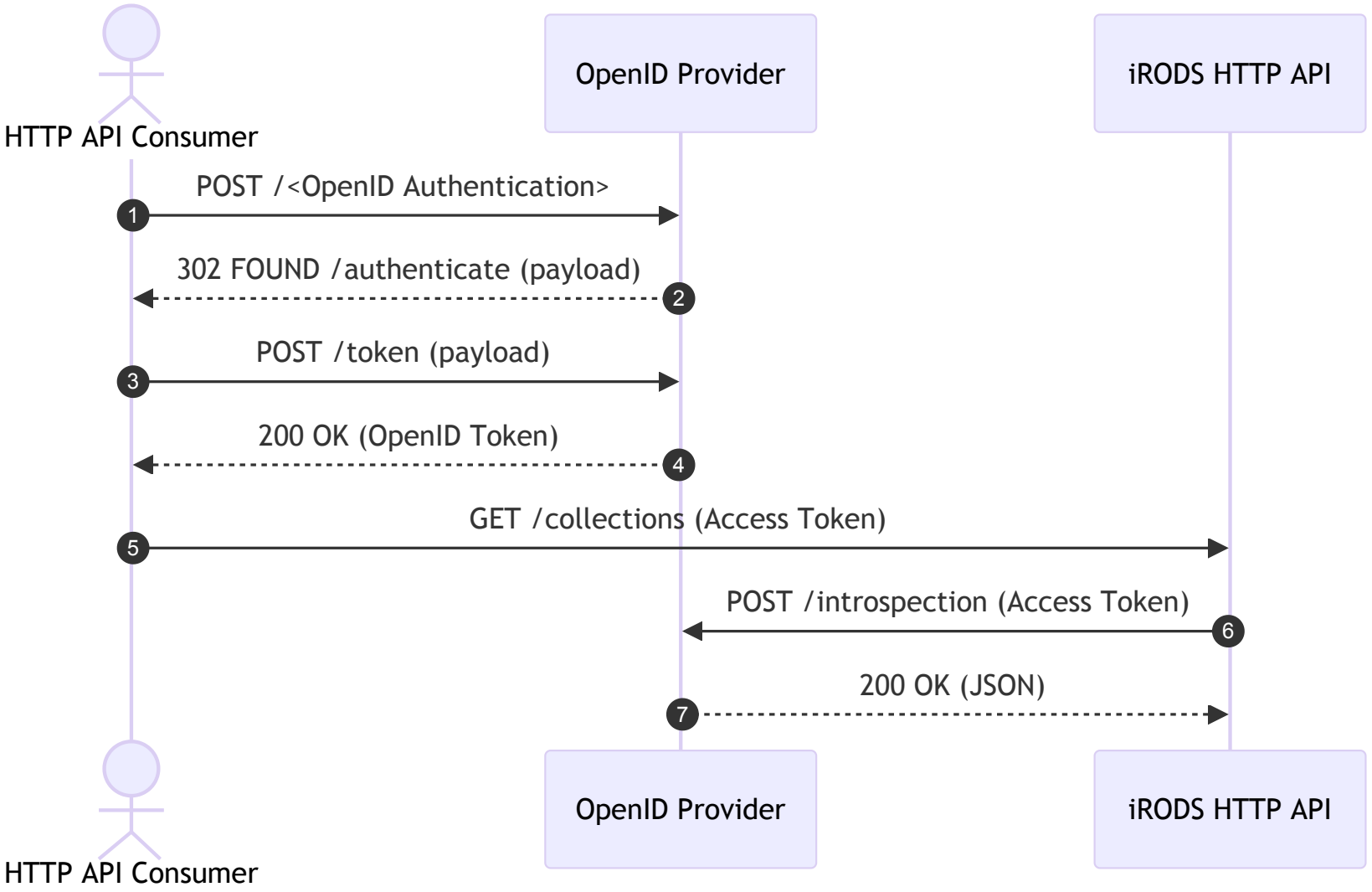
"alg" Param Value	Digital Signature or MAC Algorithm	Implementation Requirements
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS1-v1_5 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
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC performed	Optional

Table in Section 3.1 from JWA RFC 7518

Local JWT Access Token Validation - Nonstandard Behavior

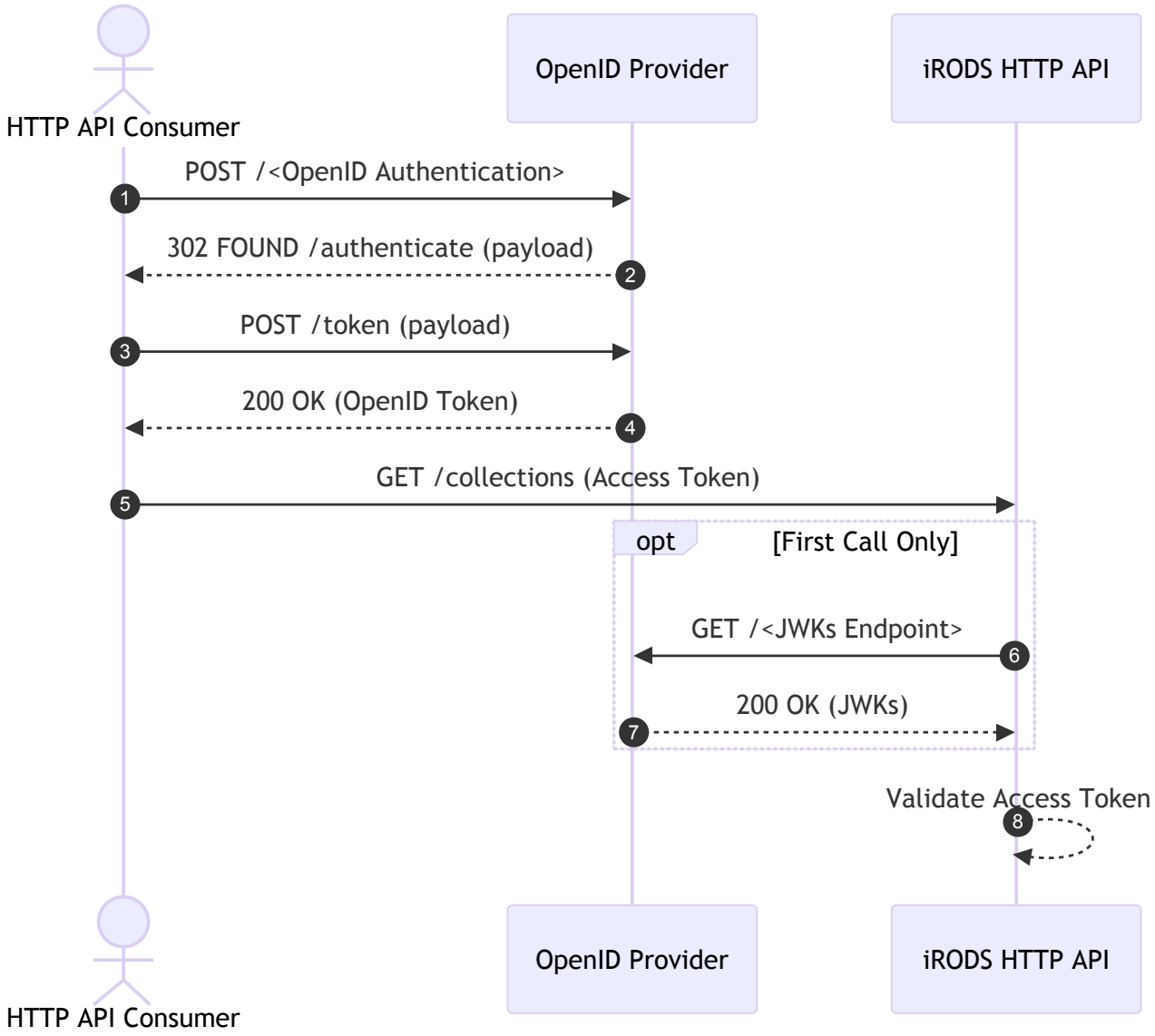
- Some OpenID providers may not use *client_secret* for symmetric ID Tokens
  - Allow for non-standard client secrets (*nonstandard_id_token_secret*)
- JWT Profile for OAuth 2.0 Access Tokens (RFC 9068)
  - Allow for "jwt" vs strictly "at+jwt" in "typ" header

HTTP API as an OAuth 2.0 Protected Resource



Example of Protected Resource Communications

HTTP API Local Validation



Example of Local Validation

HTTP API Local Validation

Demo

Future Improvements

- Encrypted JWTs
  - Depends on jwt-cpp library
- Fix timeout issue on some Providers
- Depend on OAuth 2.0 specs only
  - If enough providers support RFC 8414
- Allow configurable Access Token inspection
- Remove Client Mode
- Re-retrieve JWKs when possibly out of date
- Community suggestions

References

- OAuth 2.0
  - https://www.rfc-editor.org/rfc/rfc6749
- OpenID Connect Core
  - https://openid.net/specs/openid-connect-core-1_0.html
- OpenID Connect Client Discovery
  - http://openid.net/specs/openid-connect-discovery-1_0.html
- OAuth 2.1 Draft
  - https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-1-13
- Best Current Practice for OAuth 2.0 Security
  - https://datatracker.ietf.org/doc/html/rfc9700
- OAuth 2.0 Token Introspection
  - https://datatracker.ietf.org/doc/html/rfc7662

References

- OAuth 2.0 Authorization Server Metadata
  - https://datatracker.ietf.org/doc/html/rfc8414
- JSON Web Key (JWK)
  - https://www.rfc-editor.org/rfc/rfc7517.html
- JSON Web Algorithms (JWA)
  - https://www.rfc-editor.org/rfc/rfc7518.html
- JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens
  - https://www.rfc-editor.org/rfc/rfc9068.html
- JSON Web Encryption (JWE)
  - https://www.rfc-editor.org/rfc/rfc7516.html
- JSON Web Signature (JWS)
  - https://www.rfc-editor.org/rfc/rfc7515.html

Thank you!

Questions?