Intel Accelerated Storage Manager

Security REST API

INTEL CONFIDENTIAL

September 2017

Document Revision 1.3

Table of Contents

1. Introduction 2
2. URI structure 2
3. Error response 2
3.1 JSON Error Response Format 2
3.2 HTTP Error Response Type 3
4. Authorization methods 3
4.1 Implicit grant 3
4.1.1 Authorization request 3
4.2 Code grant 4
4.2.1 Authorization request for code grant 4
4.2.2 Authorization request for access token 5
4.3 Refreshing access token 6
4.4 Resource owner password credentials grant 7
4.4.1 Authorization request 7
5. Using access token 8
5.1 Example – Access token in query parameter 8
5.2 Example – Access token in the Authorization header 8
5.3 Error handling 8
6. Authorization Methods Flow 9
6.1 Implicit Grant Flow 9
6.2 Code Grant Flow 10
6.3 Code Grant – Token Refreshing Flow 11
6.4 Resource Owner Password Credentials Grant Flow 12

1. Introduction

This document is a reference for Security plugin API. Scope of this API covers available mechanisms of authorization in Intel ASM Service. All used mechanisms are described in RFC 6749 and RFC6750.

2. URI structure

Access to resources is provided via URI paths. Client applications should send requests over the HTTP protocol using standard GET and POST methods.

Intel ASM Service URI structure:

```
[protocol]://[host]:[port]/v1/oauth/[resource name]?[parameter name]=[parameter
value]&[parameter name]=[parameter value]
```

Where:

[protocol]	Communication protocol - http/https.
[host]	Host machine ip address or name where agent is installed.
[port]	Port number on which agent service is listening.
v1	Stands for first version of API definition.
oauth	Namespace for all resources related to commands sent to Security Plugin.
[resource name]	Name for resource to be returned.
[parameter name]	Name for URI query string parameter.
[parameter value]	Value of URI query string parameter.

3. Error response

If any request results with failure, appropriate message will be returned in JSON or HTTP format. Type of response depends on specific use case, for details, see responses for certain request.

3.1 JSON Error Response Format

Error response with JSON payload			

error	{String}	General error message	Status Code: 400 Bad Request
			Cache-Control: no-store
			Pragma: no-cache
			Content-Type: application/json
			Payload: {
			"error": "invalid_request",
			"error_description": "no scope"
			}

error_description	{String}	Detailed error message.	

3.2 HTTP Error Response Type

HTTP error response format			

For specific requests, error response can be returned in HTTP format. In this case, error information is sent back (as redirection) in query or fragment of the URI to address from 'redirect_uri' parameter of the request.

error	{String}	General error message	Status Code: 302 Found Cache-Control: no-store Pragma: no-cache
error_description	{String}	Detailed error message.	Location: https://10.10.10.10/clientapp#error=access_denied&error_description=Invalid%20credentials

4. Authorization methods

Below is a list of all available authorization methods for Security Plugin.

4.1 Implicit grant

Implicit grant is a token based authorization method dedicated for such clients like JavaScript applications (working in a user agent – typically browser) that supports redirection. It does not provide support for refresh token mechanism (described in OAuth specification).

4.1.1 Authorization request

[GET] /v1/oauth/authorize

Obtains access token if authorized.
Example: https://10.10.10.10/v1/oauth/authorize?client_id=CAPP1234&response_type=token&scope=info+rste&redirect_uri=http:%2F%2F10.10.10.11%2Fclientapp%2F&state=xyz

Request parameters (Form Data)		
client_id	{String}	Identifier of registered client application.
response_type	{String}	Must be set to **token**.
scope	{String}	List of scopes of the token separated by + in ip@scope form, where “ip” is an ip of the resource machine and “scope” is a scope name. IP is optional. Scopes are defined in plugins documentation.
redirect_uri (Optional)	{String}	Uri where user will be redirected after completing the authorization process.
state (Optional)	{String}	An opaque value used by the client to maintain state between the request and callback for preventing cross-site request forgery. Using this field is recommended.

Response Examples
Successful Authentication

Status Code: 302 Found
Cache-Control: no-store
Pragma: no-cache

```
Location:
https://10.10.10.11/clientapp#access_token=2c5e351f8651ce7ea325e9bd3cbb1d1f5c6240e15c&expires_in=3600&tok
en_type=Bearer&state=xyz
```

Authentication failed due to invalid 'client_id' or 'redirect_uri'.

```
Status Code: 400 Bad Request
Cache-Control: no-store
Pragma: no-cache
Content-Type: application/json
Payload: {
            "error": "invalid_request",
            "error_description": "Missing Client Id parameter!"
          }
```

Authentication failed from other reason then invalid 'client_id' or 'redirect_uri'.

```
Status Code: 302 Found
Location:
https://10.10.10.11/clientapp#error=access_denied&error_description=Invalid%20scope%20parameter
```

4.2 Code grant

Token based authorization type supporting refresh token mechanism. It is mostly designed for confidential applications with capability of redirection.

4.2.1 Authorization request for code grant

`[GET] /v1/oauth/authorize`

Obtains access code if authorization succeed. Access code is not yet an access token, user has to send another request to obtain a fully valid access token.
Example:

```
http://10.10.10.10/v1/oauth/authorize?client_id=CAPP1234&response_type=code&scope=info+rste&redir
ect_uri=http:%2F%2F10.10.10.11%2Fclientapp%2F&state=xyz
```

Request parameters (Form Data)

client_id	{String}	Id of registered client application.
response_type	{String}	Must be set to **code**.
scope	{String}	List of scopes of the token separated by + in ip@scope form, where “ip” is an ip of the resource machine and “scope” is a scope name. IP is optional. Scopes are defined in plugins documentation.
redirect_uri (Optional)	{String}	Uri where user will be redirected to after completing the authorization process. If redirect_uri is not present, invalid or mismatching the response will be returned in json format.
state (Optional)	{String}	An opaque value used by the client to maintain state between the request and callback for preventing cross-site request forgery. Using this field is recommended.

Response Examples

Code successfully generated (redirect_uri is present).

```
Status Code: 302 Found
Cache-Control: no-store
Pragma: no-cache
Location: https://10.10.10.11/clientapp?code=6c5e351f8651ce7ea325e9bd3cbb1d1f5c6240e15c&state=xyz
```

Authentication failed due to invalid 'client_id' or 'redirect_uri' is missing, invalid or mismatching.

```
Status Code: 400 Bad Request
Cache-Control: no-store
Pragma: no-cache
Content-Type: application/json
Payload: {
            "error": "invalid_request",
            "error_description": "Missing Client Id parameter!"
          }
```

Authentication failed from other reason then invalid 'client_id' or 'redirect_uri' ('redirect_uri' is present).

```
Status Code: 302 Found
Cache-Control: no-store
Pragma: no-cache
Location:
https://10.10.10.11/clientapp?error=access_denied&error_description=Invalid%20scope%20parameter.
```

4.2.2 Authorization request for access token

[POST] /v1/oauth/authorization_code

Gets access token based on authorization code.
Example:

https://10.10.10.10/v1/oauth/authorization_code?code=6c5e351f86513ce7ea3259bd3cbb1d1f5c6240e15&grant_type=authorization_code&client_id=CAPP1234&redirect_uri=http:%2F%2F10.10.10.11%2Fclientapp%2F&state=xyz

Request parameters

client_id	{String}	ID of client registered application.
code	{String}	Authorization code obtained from code grant request.
redirect_uri (Optional)	{String}	Uri where user will be redirected to after completing the authorization process. If redirect_uri is not present, invalid or mismatching the response will be returned in json format.
grant_type	{String}	Must be set to **authorization_code**.
state (Optional)	{String}	An opaque value used by the client to maintain state between the request and callback for preventing cross-site request forgery. Using this field is recommended.

Response Examples

Successful response with access token.

access_token	{String}	Token obtained from the authorization server.
token_type	{String}	Type of returned token. Currently only bearer is supported.
expires_in	{Number}	Token expiration time in seconds.
refresh_token	{String}	Token to be used when refreshing expired token.
scope	{String}	List of granted scopes.
domain	{String}	Domain of the authorized user account.

```
Status Code: 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
Payload: {
 "access_token": "2c4545s6d5451d1c531sd35",
 "token_type": "Bearer",
 "expires_in": 3600,
 "refresh_token": "6d46s54dc564sdc5646s65",
 "username": "user",
 "scope": "info",
 "domain": "GER",
 "state": "xyz"
}
```

username	{String}	Authorized user login.
state (Optional)	{String}	State value passed by the client.

Error Response ('redirect_uri' is present)

```
Status Code: 400 Bad Request
Cache-Control: no-store
Pragma: no-cache
Content-Type: application/json
Payload: {
  "error": "invalid_request",
  "error_description": "Missing client Id parameter!"
}
```

Error Response ('redirect_uri' is not present)

```
Status Code: 302 Found
Cache-Control: no-store
Pragma: no-cache
Location: https://10.10.10.11/clientapp?error=access_denied&error_description=Invalid%20scope%20param.
```

4.3 Refreshing access token

[POST] /v1/oauth/refresh_token

If the authorization server generates refresh token, client application makes a refresh request to the Authorization Server. Example:

https://10.10.10.10/v1/oauth/refresh_token?grant_type=refresh_token&refresh_token=5w65f4w65f465w4rwe654

Request parameters

grant_type	{String}	Must be **refresh_token.**
refresh_token	{String}	Refresh token issued to the client.
scope (Optional)	{String}	List of scopes of the token separated by + in ip@scope form, where “ip” is an ip of the resource machine and “scope” is a scope name. Scopes are defined in plugins documentation. IP is optional. Must not include any scope not originally granted by the resource owner.

Response Examples

Token refreshed successfully

access_token	{String}	Token obtained from the authorization server.
token_type	{String}	Type of returned token.
expires_in	{Number}	Token expiration time in seconds.
refresh_token	{String}	New token used when refreshing expired token.

```
Status Code: 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
Payload: {
 "access_token": "2c4545s6d5451d1c531sd35",
 "token_type": "Bearer",
 "expires_in": 3600,
 "refresh_token": "6d46s54dc564sdc5646s65"
}
```

Error Response

```
Status Code: 400 Bad Request
Cache-Control: no-store
Pragma: no-cache
```

```
Content-Type: application/json
Payload: {
  "error": "invalid_request",
  "error_description": "Missing client Id parameter!"
}
```

4.4 Resource owner password credentials grant

Grant type used when client application has special privilege or trust with authorization server. In this method, username and password are send directly to authorization server.

4.4.1 Authorization request

[POST] /v1/oauth/password_credentials

Request access token from authorization server using password credentials method.
Example:

```
https://10.10.10.10/v1/oauth/password_credentials?grant_type=password&username=markb&password=
d654d654de8&scope=info+rste
```

Request parameters (Form Data)

grant_type	{String}	Must be set to **password**.
username	{String}	User login.
password	{String}	User password.
domain (Optional)	{String}	Domain of the user account. When empty, client logs into local system account.
scope	{String}	List of scopes of the token separated by + in ip@scope form, where “ip” is an ip of the resource machine and “scope” is a scope name. IP is optional. Scopes are defined in plugins documentation.
encoded (Optional)	{Flag}	When present, password must be encoded in base64.

Response Examples

Successful Authentication

access_token	{String}	Token obtained from the authorization server.
token_type	{String}	Type of returned token.
expires_in	{Number}	Token expiration time in seconds.
refresh_token	{String}	New token used when refreshing expired token.
scope	{String}	List of granted scopes.
domain	{String}	Domain of the authorized user account.
username	{String}	Authorized user login.

```
Status Code: 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
Payload: {
 "access_token": "2c4545s6d5451d1c531sd35",
 "token_type": "Bearer",
 "expires_in": 3600,
 "refresh_token": "6d46s54dc564sdc5646s65",
 "username": "user",
 "scope": "info",
 "domain": "GER"
}
```

Error Response

```
Status Code: 400 Bad Request
Cache-Control: no-store
```

```
Pragma: no-cache
Content-Type: application/json
Payload: {
            "error": "invalid_request",
            "error_description": "Missing username parameter!"
          }
```

5. Using access token

After obtaining a valid access token, user or application should provide an access token along with a request. An access token should be included in query parameter or in the Authorization header.

5.1 Example – Access token in query parameter

```
http://10.10.10.10/v1/rste/state?access_token=2c5e351f8651ce7ea325e9bd3cbb1d1f5c6240e15c
```

After this request, the user will obtain information about a state of the plugin, if the access token is valid.

5.2 Example – Access token in the Authorization header

```
GET https://10.10.10.10/v1/rste/state
Authorization: Bearer 2c5e351f8651ce7ea325e9bd3cbb1d1f5c6240e15c
```

5.3 Error handling

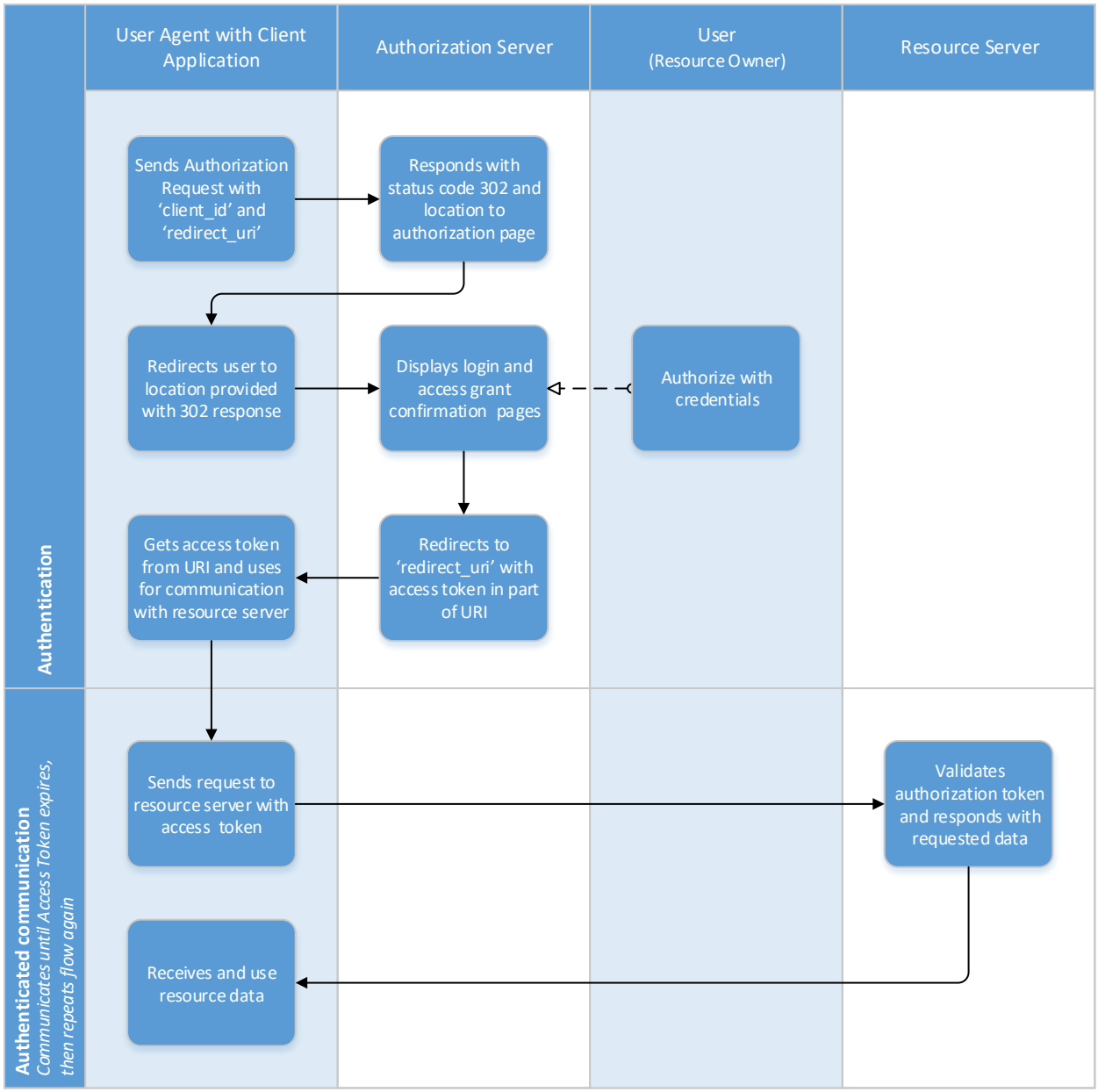
If the user will provide not existing access token, or the token that has expired already, appropriate HTTP error will be returned:

```
Status Code: 400 Bad Request
Content-Type: application/json
Payload: {
            "error": "invalid_request",
            "error_description": "Invalid token parameter!"
          }
```

6. Authorization Methods Flow

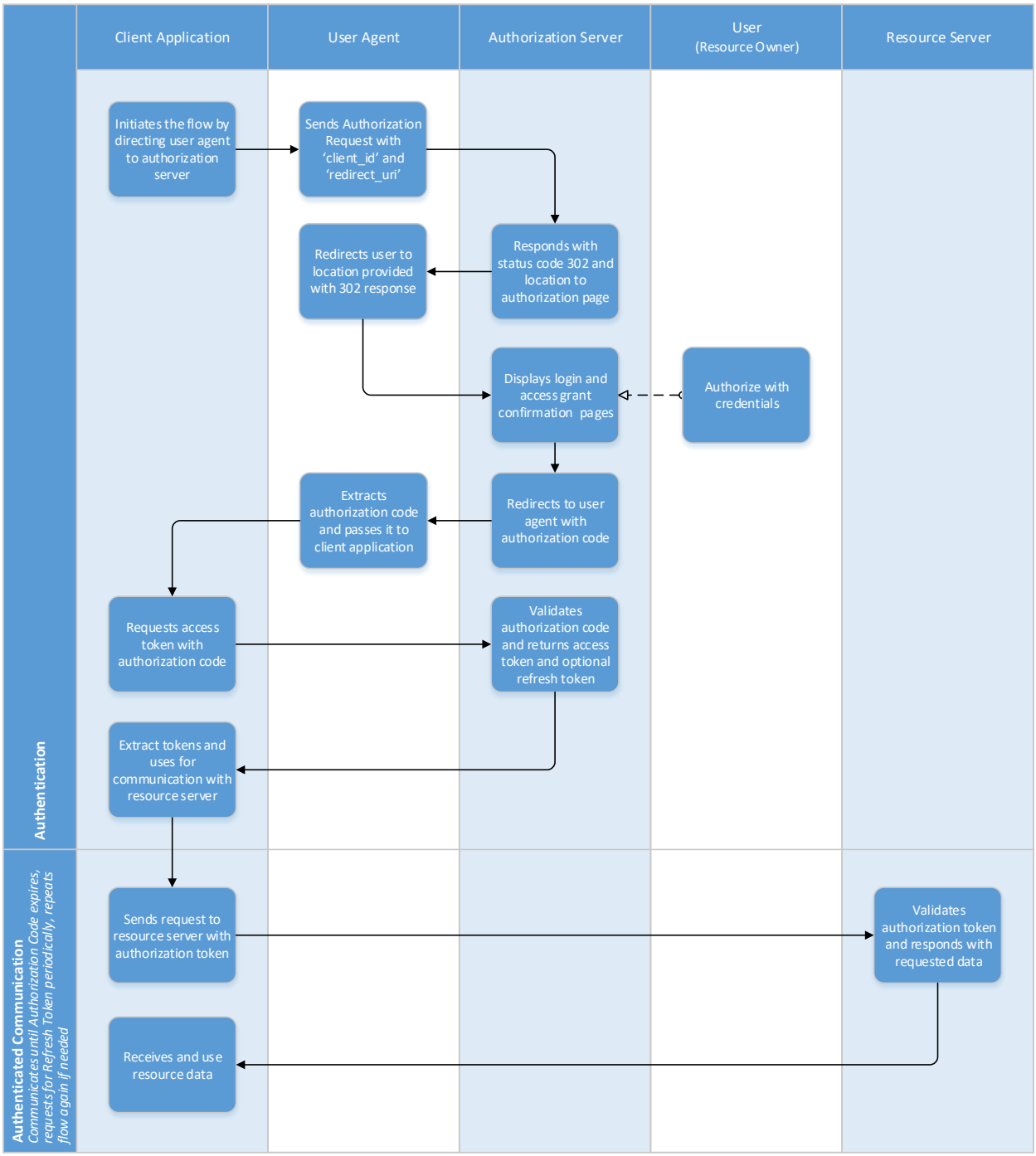
6.1 Implicit Grant Flow

Detailed information about this grant type can be found in RFC6749 in section 4.2.



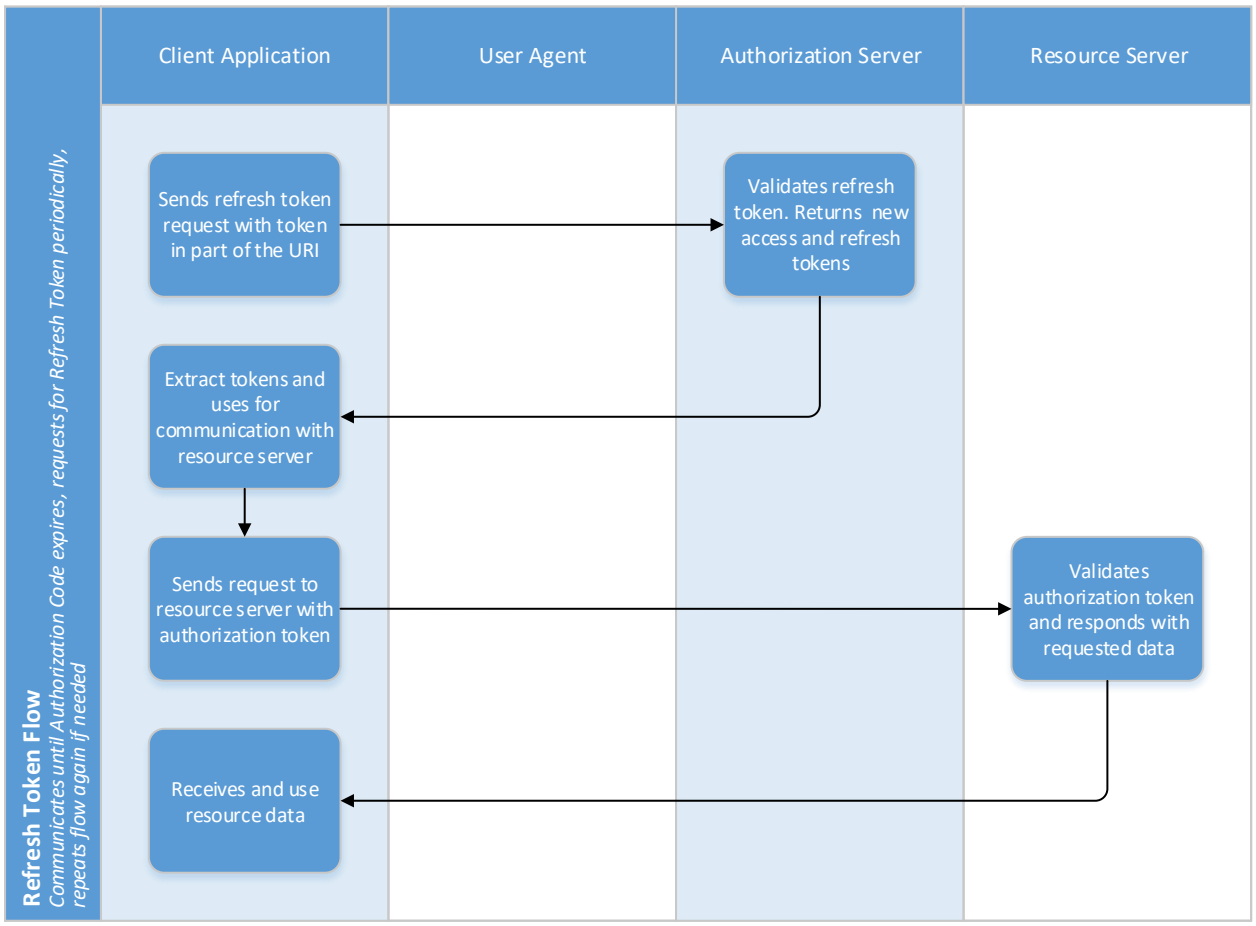
6.2 Code Grant Flow

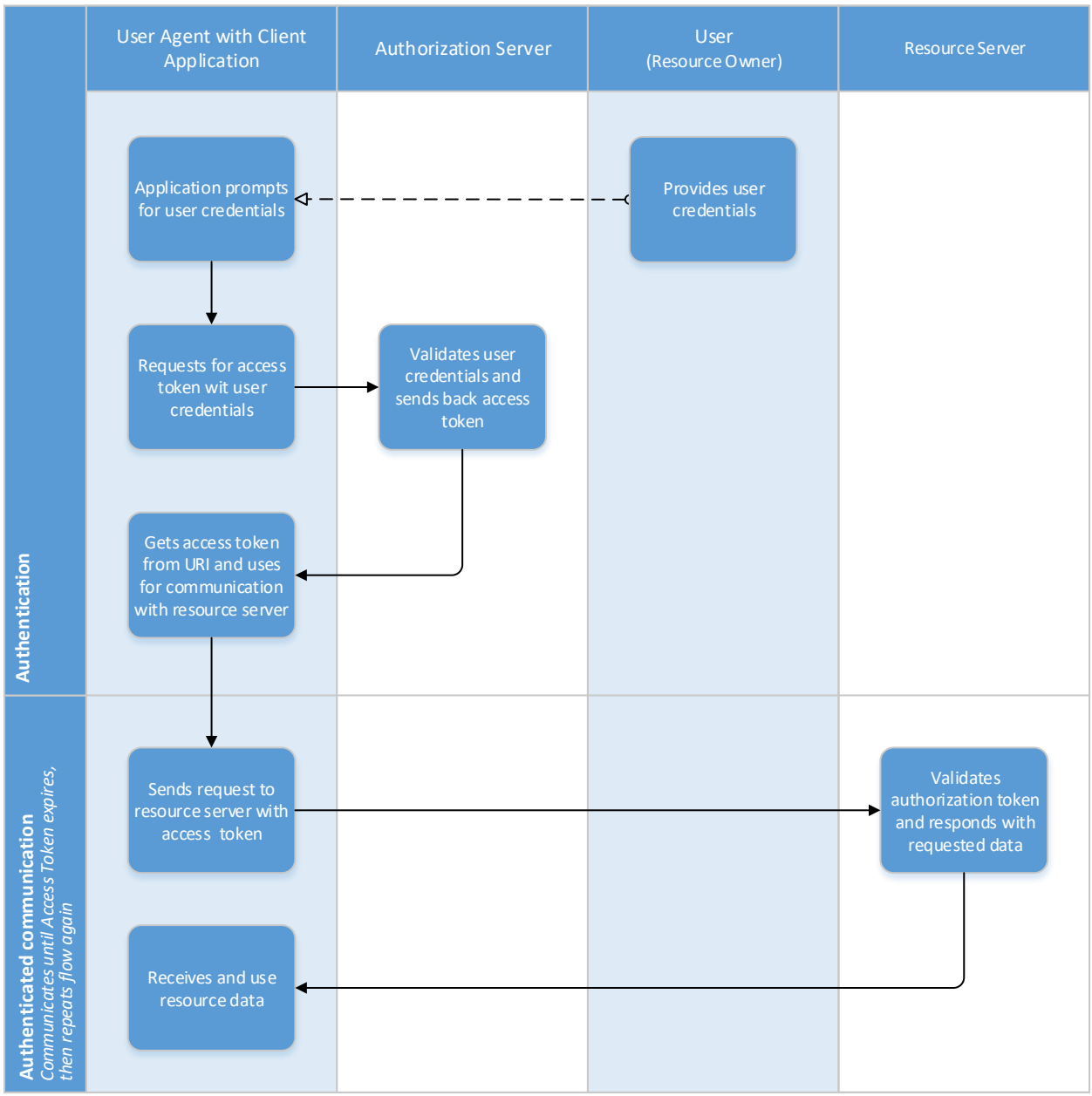
Detailed information about this grant type can be found in RFC6749 in section 4.1.



6.3 Code Grant – Token Refreshing Flow

Detailed information about refreshing token can be found in RFC6749 in section 6.



6.4 Resource Owner Password Credentials Grant Flow

Detailed information about this grant type can be found in RFC6749 in section 4.3.