HMISLynk Platform OAuth 2.0 Documentation

Overview

HMISLynk Platform uses OAuth 2.0 open-standard protocol to allow a user to authorize a Trusted App (aka client) to access data stored in the HMISLynk, after he/she has been authenticated (logged-in).

A third-party application or a Trusted App (aka client) often requires limited access to a user's data stored in the HMISLynk platform. To ensure that access to data stored in the HMISLynk is not abused, all requests for access must be approved by the user.

OAuth 2.0 is a relatively simple protocol and a developer can integrate with HMISLynk Platform's OAuth 2.0 endpoints without too much effort. In a nutshell, a Trusted App developer registers his/her application (aka client) with HMISLynk, redirects a browser to a URL, parses a token from the response, and uses the token to invoke a HMISLynk API.

HMISLynk Platform supports the OAuth 2.0 protocol with bearer tokens for web and native (aka installed) applications.

Trusted Apps (aka Clients) follow the same basic steps when accessing a HMISLynk API using OAuth 2.0. At a high level, using OAuth 2.0 to access a HMISLynk API consists of the following four steps:

1 Register a Trusted App (aka Client)

All Trusted Apps (aka clients) that access a HMISLynk API must be registered. The result of this registration process is a set of values (e.g. client_id, client_secret, redirect_uri, list of APIs to be accessed, etc.) that are known to both HMISLynk and the Trusted App. The set of values needed varies based on the type of application being built. For example a JavaScript application does not require a secret, but a server-side web application or a native application does. To register as a Trusted App developer, please create an account at HMIS Developer Network and follow the steps as guided.

2 Obtain an Access Token from the HMISLynk authorization service

Before a Trusted App (aka client) can access HMISLynk APIs, it must obtain an access token that grants access to the APIs.

There are several ways to make this request, and they vary based on the type of Trusted App.

The request requires the user to login to HMISLynk. After logging in, the user will see the permissions (list of APIs to be accessed) requested by the application and is asked if he/she is willing to grant the application those permissions. This process is called "user consent".

If the user grants permission to the Trusted App, the Trusted App will be sent an access token or an authorization code (which is used to obtain an access token). If the user does not grant permission to the application, the HMISLynk Authorization Service returns an error.

3 Send Access Token to an API

After the Trusted App (aka client) has obtained an access token, it may send the access token in a request to a HMISLynk API. The Access token is sent to a HMISLynk API in the HTTP Authorization header as shown below.

Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA

4 Refresh the Access Token (optional)

Access tokens have a limited lifetime and, in some cases, an application (aka client) needs access to HMISLynk APIs beyond the lifetime of a single access token. When this is the case, the application can obtain what is called a refresh token. A refresh token allows the application to obtain new access tokens.

Authorization Flows

HMISLynk Platform supports the following OAuth 2.0 authorization flows:

- 1 Authorization Code Grant Flow
  Server-Side Web Applications and **Native Applications** (i.e. iPhone / iPad apps, Android apps) should use this flow.

- 2 Implicit Grant Flow
  Browser-based **Client-Side Web Applications** (using JavaScript, Flash, etc.) should use this flow.

1 Authorization Code Grant Flow

1.0 Overview

Server-Side Web Applications and **Native Applications** (i.e. iPhone / iPad apps, Android apps) should use this flow.

Step (A): The Trusted App (aka client) initiates this flow by directing user's browser to HMISLynk's **authorization endpoint** URL. The Trusted App (aka client) includes its client identifier, local state, and a redirection URI to which the HMISLynk authorization service will send the browser back once access is granted (or denied).

In order for a Trusted App (aka client) to be approved, it must implement the OAuth workflow by launching the mobile device's **default browser, external to the application**. (The use of an embedded browser is not acceptable.)

Step (B): The HMISLynk authorization service authenticates the user (via the browser). After authentication (logging in), the user will see the permissions (list of APIs to be accessed) requested by the application and is asked if he/she is willing to grant the application those permissions. (Consent process)

Step (C): Assuming the user grants access, the HMISLynk authorization service redirects the browser back to the Trusted App (aka client) using the redirection URI provided earlier. The redirection URI includes an authorization code and any local state provided by the application (aka client) earlier.

Step (D): The Trusted App (aka client) requests an access token from the HMISLynk's **token endpoint** by including the authorization code received in the previous step. When making the request, the Trusted App (aka client) authenticates with the HMISLynk authorization service by including its credential (**i.e. client_id and client_secret**). The Trusted App (aka client) also includes the redirection URI used to obtain the authorization code for verification.

Step (E): The HMISLynk authorization service authenticates the Trusted App (aka client), validates the authorization code, and ensures the redirection URI received matches the URI used to redirect the application (aka client) in step (C). If valid, the HMISLynk authorization service responds back with an access token and optionally, a refresh token. If a refresh token is present in the response, then the application (aka client) may use it to obtain new access tokens at any time.

The application (aka client) can invoke/access a HMISLynk API after it receives the access token.

1.1 Direct the user's browser to HMISLynk's authorization endpoint URL

The application (aka client) initiates this flow by directing user's browser to HMISLynk's **authorization endpoint** URL with a set of query string parameters.

Icon

In order for a Trusted App (aka client) to be approved, it must implement the OAuth workflow by launching the device's **default browser, external to the Trusted App**. (The use of an embedded browser is not acceptable.)

The set of query string parameters supported by the HMISLynk authorization service for Authorization Code Grant Flow are:

Parameter Name	Parameter Value	Description	Required?
response_type	**code**	For Authorization Code Grant Flow, the value of this parameter must be **code**.	Yes
client_id	The client identifier issued to the Trusted App (aka client) during the Trusted App / client registration process.	Indicates the Trusted App (aka client) that is making the request.	Yes
redirect_uri	One of the redirect_uri values registered during the Trusted App / client registration process.	Determines where the response is sent. The value of this parameter must exactly match one of the values registered during the Trusted App / client registration process (including the http or https schemes, case, and trailing '/').	Yes
state	any string	An opaque value used by the Trusted App (aka client) to maintain state between the request and callback. The	Yes

		HMISLynk authorization service includes this value when redirecting the browser back to the Trusted App (aka client). The parameter should be used for preventing cross-site request forgery as described in OAuth Section 10.12	
access_type	**online** or **offline**	Indicates if the Trusted App (aka client) needs to access a HMISLynk API when the user is not present at the browser. This parameter defaults to **online**. If a Trusted App(aka client) needs to refresh access tokens when the user is not present at the browser, then use **offline**. This will result in client obtaining a refresh token the first time the client exchanges an authorization code for a user.	No
approval_prompt	**force** or **auto**	Indicates if the user should be re-prompted for consent. The default is **auto**, so a given user should only see the consent page the first time through the sequence. If the value is **force**, then the user sees a consent page even if they have previously given consent to the client.	No

An example URL is shown below.

https://www.hmislynk.com/authorization-service/oauth/authorize?
response_type=code
&client_id=YOUR_TRUSTED_APP_ID
&redirect_uri=YOUR_REDIRECT_URI
&state=SOME_ARBITRARY_BUT_UNIQUE_STRING

For security, the value of **redirect_uri** parameter must exactly match one of the values registered during the Trusted App (aka client) registration process (including the http or https schemes, case, and trailing '/').

The **state** parameter should be set to some arbitrary string you generate uniquely for each auth request. This value will be passed back as a parameter to the **redirect_uri** once the user has authorized the Trusted App (aka client) and the Trusted App should check that the returned value matched the value it passed in at the start of the flow. This guards against Cross-site Request

Forgery by ensuring the incoming redirect is part of the auth flow initiated by the Trusted App (aka client).

1.2 The user is prompted to authorize the Trusted App (aka client)

If the user is not logged-in into HMISLynk, He/She will be prompted (via the browser) to log-in.

If the user has not already authorized the Trusted App, He/She will be prompted (via the browser) to authorize the Trusted App. This process is called "user consent".

1.3 The user's browser is redirected back to the Trusted App (aka client)

If the user grants the access request, the HMISLynk authorization service issues an authorization code and delivers it to the Trusted App (aka client) by adding the following parameters to the query component of the redirection URI using the "application/x-www-form-urlencoded" format:

Parameter Name	Description / Parameter Value	Required?
code	The authorization code generated / issued by the HMISLynk authorization service. The authorization code expires shortly (max lifetime of 10 minutes) after it is issued to mitigate the risk of leaks. The Trusted App (aka client) must not use the authorization code more than once.	Yes
state	The exact value received from the Trusted App (aka client).	Yes if the state parameter was included in the authorization request

For example, the HMISLynk authorization service redirects the browser by sending the following HTTP response:

HTTP/1.1 302 Found
Location:
YOUR_REDIRECT_URI?code=AUTHORIZATION_CODE_GENERATED_BY_HMIS_LYNK&state=YOUR_STATE_VALUE

In future, additional parameters/fields may be included in the response. The client must be coded to ignore the parameters/fields it does not recognize.

If the request fails due to a missing, invalid, or mismatching redirection URI, or if the client identifier is missing or invalid, the HMISLynk authorization service will inform the user of the error, and will not automatically redirect the browser to the invalid redirection URI.

If the user denies the access request or if the request fails for reasons other than a missing or invalid redirection URI, the HMISLynk authorization service informs the Trusted App (aka client) by adding the following parameters to the query component of the redirection URI using the "application/x-www-form-urlencoded" format:

Parameter Name	Description / Parameter Value	Required?
error	A single error code from the following: **invalid_request** The request is missing a required parameter, includes an invalid parameter value, or is otherwise malformed. **unauthorized_client** The Trusted App (aka client) is not authorized to request an authorization code using this method. **access_denied** The user or HMISLynk authorization service denied the request. **unsupported_response_type** The HMISLynk authorization service does not support the specified response type. **server_error** The HMISLynk authorization service encountered an unexpected condition which prevented it from fulfilling the request. **temporarily_unavailable** The HMISLynk authorization service is currently unable to handle the request due to a temporary overloading or maintenance of the server.	Yes
error_description	A human-readable UTF-8 encoded text providing additional information, used to assist the Trusted App (aka client) developer in understanding the error that occurred.	No
state	The exact value received from the Trusted App (aka client).	Yes if a **state** parameter was included in the authorization request.

For example, the HMISLynk authorization service redirects the browser by sending the following HTTP response:

HTTP/1.1 302 Found

Location:
YOUR_REDIRECT_URI?error=ERROR_CODE_RETURNED_BY_HMIS_LYNK&state=YOUR_STATE_VALUE

1.4 Exchange the authorization code for an access token

After the Trusted App (aka client) receives the authorization code, it should make an HTTPs POST request to HMISLynk's **token endpoint** to exchange the authorization code for an access token and a refresh token.

The Trusted App (aka client) makes a request to HMISLynk's **token endpoint** by adding the following parameters using the "Trusted App/x-www-form-urlencoded" format in the HTTP request entity-body:

Parameter Name	Description / Parameter Value	Required?
grant_type	As defined in the OAuth 2.0 specification, the value of this parameter must be set to **authorization_code**.	Yes
code	The authorization code generated / issued by the HMISLynk authorization service.	Yes
redirect_uri	The **redirect_uri** parameter that was included in the initial request (for authorization code), and their values must be identical.	Yes

The Trusted App (aka client) must specify its credentials (i.e. client id/client secret issued during Trusted App/client registration) using HTTP Basic authentication scheme as defined in RFC2617. The client identifier is used as the username, and the client secret is used as the password.

An example request

POST authorization-service/oauth/token HTTP/1.1
Host: www.hmislynk.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: Trusted App/x-www-form-urlencoded;charset=UTF-8
Accept: application/json
Accept-Charset: utf-8

grant_type=authorization_code&

code=AUTHORIZATION_CODE_GENERATED_BY_HMIS_LYNK&
redirect_uri=YOUR_REDIRECT_URI

If the access token request is valid and authorized, the HMISLynk authorization service issues an access token and optional refresh token, and constructs the response by adding the following parameters to the entity body of the HTTP response with a 200 (OK) status code:

Parameter Name	Description / Parameter Value	Required?
access_token	The access token generated / issued by the HMISLynk authorization service.	Yes
token_type	Indicates the type of token returned. At this time, this field will always have the value **Bearer**. The Value is case insensitive.	Yes
expires_in	The lifetime in seconds of the access token. For example, the value "3600" denotes that the access token will expire in one hour from the time the response was generated.	Yes
refresh_token	A refresh token which can be used to obtain a new access token. Refresh tokens are valid until the user revokes access. This field is only present if **access_type=offline** is included in the initial request (for authorization code)	No

The parameters are included in the entity body of the HTTP response using the "application/json" media type as defined by RFC4627. The parameters are serialized into a JSON structure by adding each parameter at the highest structure level. Parameter names and string values are included as JSON strings. Numerical values are included as JSON numbers. The order of parameters does not matter and can vary.

The HMISLynk authorization service includes the HTTP "Cache-Control" response header field RFC2616 with a value of "no-store" as well as the "Pragma" response header field RFC2616 with a value of "no-cache".

An example successful response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "access_token":"ACESS_TOKEN_GENERATED_BY_HMIS_LYNK",
   "token_type":"Bearer",
```

```
  "expires_in":NUMBER_OF_SECONDS_UNTIL_ACCESS_TOKEN_EXPIRES,
  "refresh_token":"REFRESH_TOKEN_GENERATED_BY_HMIS_LYNK"
}
```

In future, additional parameters/fields may be included in the response. The client must be coded to ignore the parameters/fields it does not recognize.

If the request is invalid or application (aka client) authentication failed, the HMISLynk authorization service responds with an HTTP 400 (Bad Request) status code and includes the following parameters with the response:

Parameter Name	Description / Parameter Value	Required?
error	A single error code from the following: **invalid_request** The request is missing a required parameter, includes an unsupported parameter value (other than grant type), repeats a parameter, includes multiple credentials, utilizes more than one mechanism for authenticating the client, or is otherwise malformed. **invalid_client** Client authentication failed (e.g. unknown client, no client authentication included, or unsupported authentication method). **invalid_grant** The provided authorization grant (e.g. authorization code, resource owner credentials) or refresh token is invalid, expired, revoked, does not match the redirection URI used in the authorization request, or was issued to another client. This error code is also returned if the account is disabled. **unauthorized_client** The authenticated Trusted App (aka client) is not authorized to use the specified grant type. **unsupported_grant_type** The specified grant type is not supported by the HMISLynk authorization service. **server_error** The HMISLynk authorization service encountered an unexpected condition which prevented it from fulfilling the request. **temporarily_unavailable** The HMISLynk authorization service is currently unable to handle the request due to a temporary overloading or maintenance of the server.	Yes

error_description	A human-readable UTF-8 encoded text providing additional information, used to assist the Trusted App (aka client) developer in understanding the error that occurred.	No

An example error response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
"error":"ERROR_CODE_RETURNED_BY_HMIS_LYNK"
}
```

1.5 Calling a HMISLynk API

After the application (aka client) has obtained an access token, the Trusted App can access a HMISLynk API by including it in an **Authorization: Bearer** HTTP header as shown below:

```
Authorization: Bearer ACCESS_TOKEN_GOES_HERE
```

1.6 Using a Refresh Token

If the HMISLynk authorization service issued a refresh token to the Trusted App (aka client), the Trusted App (aka client) can use the refresh token to obtain a new access token any time until the the refresh token is revoked.

The Trusted App (aka client) makes an HTTPs POST request to HMISLynk's **token endpoint** by adding the following parameters using the "application/x-www-form-urlencoded" format in the HTTP request entity-body:

Parameter Name	Description / Parameter Value	Required?
grant_type	As defined in the OAuth 2.0 specification, the value of this parameter must be set to **refresh_token**.	Yes
refresh_token	The refresh token issued to the Trusted App (aka client).	Yes

The application (aka client) must specify its credentials (i.e. client id/client secret issued during client registration) using HTTP Basic authentication scheme as defined in RFC2617. The client identifier is used as the username, and the client secret is used as the password.

An example request:

```
POST authorization-service/oauth/token HTTP/1.1
Host: www.hmislynk.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Accept: application/json
Accept-Charset: utf-8

grant_type=refresh_token&
refresh_token=REFRESH_TOKEN_GENERATED_BY_HMIS_LYNK
```

If the request is valid and authorized, the HMISLynk authorization service issues an access token, and constructs the response by adding the following parameters to the entity body of the HTTP response with a 200 (OK) status code:

Parameter Name	Description / Parameter Value	Required?
access_token	The access token generated / issued by the HMISLynk authorization service.	Yes
token_type	Indicates the type of token returned. At this time, this field will always have the value **Bearer**. The Value is case insensitive.	Yes
expires_in	The lifetime in seconds of the access token. For example, the value "3600" denotes that the access token will expire in one hour from the time the response was generated.	Yes

The parameters are included in the entity body of the HTTP response using the "application/json" media type as defined by RFC4627. The parameters are serialized into a JSON structure by adding each parameter at the highest structure level. Parameter names and string values are included as JSON strings. Numerical values are included as JSON numbers. The order of parameters does not matter and can vary.

The HMISLynk authorization service includes the HTTP "Cache-Control" response header field RFC2616 with a value of "no-store" as well as the "Pragma" response header field RFC2616 with a value of "no-cache".

An example successful response:

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "access_token":"ACESS_TOKEN_GENERATED_BY_HMIS_LYNK",
   "token_type":"Bearer",
   "expires_in":NUMBER_OF_SECONDS_UNTIL_ACCESS_TOKEN_EXPIRES
}
```

In future, additional parameters/fields may be included in the response. The client must be coded to ignore the parameters/fields it does not recognize.

If the request is invalid or application (aka client) authentication failed, the HMISLynk authorization service responds with an HTTP 400 (Bad Request) status code and includes the following parameters with the response:

Parameter Name	Description / Parameter Value	Required?
error	A single error code from the following: **invalid_request** The request is missing a required parameter, includes an unsupported parameter value (other than grant type), repeats a parameter, includes multiple credentials, utilizes more than one mechanism for authenticating the application (aka client), or is otherwise malformed. **invalid_client** Application (aka client) authentication failed (e.g. unknown client, no client authentication included, or unsupported authentication method). **invalid_grant** The provided authorization grant (e.g. authorization code, resource owner credentials) or refresh token is invalid, expired, revoked, does not match the redirection URI used in the authorization request, or was issued to another client. This error code is also returned if the account is disabled. **unauthorized_client** The authenticated application (aka client) is not authorized to use the specified grant type. **unsupported_grant_type** The specified grant type is not supported by the HMISLynk authorization service. **server_error**	Yes

	The HMISLynk authorization service encountered an unexpected condition which prevented it from fulfilling the request. **temporarily_unavailable** The HMISLynk authorization service is currently unable to handle the request due to a temporary overloading or maintenance of the server.	
error_description	A human-readable UTF-8 encoded text providing additional information, used to assist the application (aka client) developer in understanding the error that occurred.	No

An example error response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
"error":"ERROR_CODE_RETURNED_BY_HMIS_LYNK"
}
```

1.7 Revoking an Access Token or a Refresh Token

An application (aka client) uses the HMISLynk's **revocation endpoint** to revoke an access token or a refresh token (i.e. refresh token and all related access tokens).

Developers may use this feature when configuring a "Log Out" button in their application.

The application (aka client) makes a request to the HMISLynk's **revocation endpoint** by adding "one" of following parameters using the "application/x-www-form-urlencoded" format in the HTTP request entity-body:

Parameter Name	Description / Parameter Value	Required?
access_token	The access token to be revoked.	One of the **access_token** or **refresh_token** is required.
refresh_token	The refresh token to be revoked. In this case, all related access tokens are revoked as well.	One of the **access_token** or **refresh_token** is required.

An example request for revoking an access token

```
POST authorization-service/oauth/revoke HTTP/1.1
Host: www.hmislynk.com
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Accept: application/json
Accept-Charset: utf-8

access_token=ACCESS_TOKEN_GENERATED_BY_HMIS_LYNK
```

An example request for revoking a refresh token:

```
POST authorization-service/oauth/revoke HTTP/1.1
Host: www.hmislynk.com
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Accept: application/json
Accept-Charset: utf-8

refresh_token=REFRESH_TOKEN_GENERATED_BY_HMIS_LYNK
```

The HMISLynk authorization service indicates successful processing of the request by returning an HTTP status code 200 with the access/refresh token that has been revoked successfully.

An example response if the specified access token has been revoked successfully:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "access_token":"ACCESS_TOKEN_GENERATED_BY_HMIS_LYNK"
}
```

An example response if the specified refresh token has been revoked successfully:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "refresh_token":"REFRESH_TOKEN_GENERATED_BY_HMIS_LYNK"
}
```

In future, additional parameters/fields may be included in the response. The client must be coded to ignore the parameters/fields it does not recognize.

For all error conditions, a status code 400 is used along with one of the following error responses.

Parameter Name	Description / Parameter Value	Required?
error	A single error code from the following: **invalid_request** The request is missing a required parameter, includes an unsupported parameter value, repeats a parameter, includes multiple tokens, or is otherwise malformed. **server_error** The HMISLynk authorization service encountered an unexpected condition which prevented it from fulfilling the request. **temporarily_unavailable** The HMISLynk authorization service is currently unable to handle the request due to a temporary overloading or maintenance of the server.	Yes
error_description	A human-readable UTF-8 encoded text providing additional information, used to assist the application (aka client) developer in understanding the error that occurred.	No

An example error response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
"error":"ERROR_CODE_RETURNED_BY_HMIS_LYNK"
}
```

2 Implicit Grant Flow

2.0 Overview

Browser-based **Client-Side Web Applications** (using JavaScript, Flash, etc.) should use this flow.

The Implicit Grant Flow is used to obtain access tokens (it does not support the issuance of refresh tokens) and is optimized for applications (aka clients) which are typically implemented in a browser using a scripting language such as JavaScript.

Unlike the Authorization Code Grant Flow in which the application (aka client) makes separate requests for authorization code and access token, the client receives the access token as the result of the authorization request.

The Implicit Grant Flow does not include application (aka client) authentication, and relies on the presence of the HMISLynk user and the registration of the redirection URI.

Step (A): The application (aka client) initiates this flow by directing user's browser to HMISLynk's **authorization endpoint** URL. The application (aka client) includes its client identifier, local state, and a redirection URI to which the HMISLynk authorization service will send the browser back once access is granted (or denied).

Icon

In order for an application (aka client) to be approved, it must implement the OAuth workflow by launching the device's **default browser, external to the application**. (The use of an embedded browser is not acceptable.)

Step (B): The HMISLynk authorization service authenticates the user (via the browser). After authentication (logging in), the user will see the permissions (list of APIs to be accessed) requested by the application and is asked if he/she is willing to grant the application those permissions.

Step (C): Assuming the user grants access, the HMISLynk authorization service redirects the browser back to the application (aka client) using the redirection URI provided earlier. The redirection URI includes an access token and any local state provided by the application (aka client) earlier in the URI fragment.

Step (D): The browser follows the redirection instructions by making a request to the web-hosted client resource (which does not include the fragment per RFC2616). The browser retains the fragment information locally.

Step (E): The web-hosted client resource returns a web page (typically an HTML document with an embedded script) capable of accessing the full redirection URI including the fragment retained by the browser, and extracting the access token (and other parameters) contained in the fragment.

Step (F): The browser executes the script provided by the web-hosted client resource locally, which extracts the access token and passes it to the application (aka client).

The application (aka client) can invoke/access a HMISLynk API after it receives the access token.

2.1 Direct the user's browser to HMISLynk's authorization endpoint URL

The application (aka client) initiates this flow by directing user's browser to HMISLynk's **authorization endpoint** URL with a set of query string parameters.

Icon

In order for an application (aka client) to be approved, it must implement the OAuth workflow by launching the device's **default browser, external to the application**. (The use of an embedded browser is not acceptable.)

The set of query string parameters supported by the HMISLynk authorization service for Implicit Grant Flow are:

Parameter Name	Parameter Value	Description	Required?
response_type	**token**	For Implicit Grant Flow, the value of this parameter must be **token**.	Yes
client_id	The client identifier issued to the application (aka client) during the application / client registration process.	Indicates the application (aka client) that is making the request.	Yes
redirect_uri	One of the redirect_uri values registered during the application / client registration process.	Determines where the response is sent. The value of this parameter must exactly match one of the values registered during the application / client registration process (including the http or https schemes, case, and trailing '/').	Yes
state	any string	An opaque value used by the application (aka client) to maintain state between the request and callback. The HMISLynk authorization service includes this value when redirecting the browser back to the application (aka client). The parameter should be used for preventing cross-site request forgery as described in OAuth Section 10.12	Yes
approval_prompt	**force** or **auto**	Indicates if the user should be re-prompted for consent. The default is **auto**, so a given user should only see the consent page the first time through the sequence. If the value is **force**, then the user sees a consent page even if they have previously given consent to the client.	No

An example URL is shown below.

https://www.hmislynk.com/authorization-service/oauth/authorize?
response_type=token
&client_id=YOUR_CLIENT_ID
&redirect_uri=YOUR_REDIRECT_URI
&state=SOME_ARBITRARY_BUT_UNIQUE_STRING

For security, the value of **redirect_uri** parameter must exactly match one of the values registered during the application (aka client) registration process (including the http or https schemes, case, and trailing '/').

The **state** parameter should be set to some arbitrary string you generate uniquely for each auth request. This value will be passed back as a parameter to the **redirect_uri** once the user has authorized the application (aka client) and the application should check that the returned value matched the value it passed in at the start of the flow. This guards against Cross-site Request Forgery by ensuring the incoming redirect is part of the auth flow initiated by the application (aka client).

2.2 The user is prompted to authorize the application (aka client)

If the user is not logged-in into HMISLynk, He/She will be prompted (via the browser) to log-in.

If the user has not already authorized the application, He/She will be prompted (via the browser) to authorize the application. This process is called "user consent".

2.3 The user's browser is redirected back to the application (aka client)

If the user grants the access request, the HMISLynk authorization service issues an access token and delivers it to the application (aka client) by adding the following parameters to the fragment component of the redirection URI using the "application/x-www-form-urlencoded" format:

Parameter Name	Description / Parameter Value	Required?
access_token	The access token generated / issued by the HMISLynk authorization service.	Yes
token_type	Indicates the type of token returned. At this time, this field will always have the value **Bearer**. The Value is case insensitive.	Yes
expires_in	The lifetime in seconds of the access token. For example, the value "3600" denotes that the access token will expire in one hour from the time the response was generated.	Yes

state	The exact value received from the application (aka client).	Yes if the **state** parameter was included in the authorization request

Since a fragment is not returned by the browser to the application (aka client), client-side script must parse the fragment and extract the value of the access_token parameter.

For example, the HMISLynk authorization service redirects the browser by sending the following HTTP response:

HTTP/1.1 302 Found
Location:
YOUR_REDIRECT_URI#access_token=ACESS_TOKEN_GENERATED_BY_HMIS_LYNK&token_type=Bearer

&expires_in=NUMBER_OF_SECONDS_UNTIL_ACCESS_TOKEN_EXPIRES&state=YOUR_STATE_VALUE

In future, additional parameters/fields may be included in the response. The client must be coded to ignore the parameters/fields it does not recognize.

If the request fails due to a missing, invalid, or mismatching redirection URI, or if the client identifier is missing or invalid, the HMISLynk authorization service will inform the user of the error, and will not automatically redirect the browser to the invalid redirection URI.

If the user denies the access request or if the request fails for reasons other than a missing or invalid redirection URI, the HMISLynk authorization service informs the application (aka client) by adding the following parameters to the fragment component of the redirection URI using the "application/x-www-form-urlencoded" format:

Parameter Name	Description / Parameter Value	Required?
error	A single error code from the following: **invalid_request** The request is missing a required parameter, includes an invalid parameter value, or is otherwise malformed. **unauthorized_client** The application (aka client) is not authorized to request an access token using this method. **access_denied** The user or HMISLynk authorization service denied	Yes

	the request. **unsupported_response_type** The HMISLynk authorization service does not support the specified response type. **server_error** The HMISLynk authorization service encountered an unexpected condition which prevented it from fulfilling the request. **temporarily_unavailable** The HMISLynk authorization service is currently unable to handle the request due to a temporary overloading or maintenance of the server.	
error_description	A human-readable UTF-8 encoded text providing additional information, used to assist the application (aka client) developer in understanding the error that occurred.	No
state	The exact value received from the application (aka client).	Yes if a **state** parameter was included in the authorization request.

For example, the HMISLynk authorization service redirects the browser by sending the following HTTP response:

HTTP/1.1 302 Found
Location:
YOUR_REDIRECT_URI#error=ERROR_CODE_RETURNED_BY_HMIS_LYNK&state=YOUR_STATE_VALUE

2.4 Calling a HMISLynk API

After the application (aka client) has obtained an access token, the application can access a HMISLynk API by including it in an **Authorization: Bearer** HTTP header as shown below:

Authorization: Bearer ACCESS_TOKEN_GOES_HERE

2.5 Revoking an Access Token

An application (aka client) uses the HMISLynk's **revocation endpoint** to revoke an access token.

Developers may use this feature when configuring a "Log Out" button in their application.

The application (aka client) makes a request to the HMISLynk's **revocation endpoint** by adding the following parameter using the "application/x-www-form-urlencoded" format in the HTTP request entity-body:

Parameter Name	Description / Parameter Value	Required?
access_token	The access token to be revoked.	Yes

An example request for revoking an access token

```
POST authorization-service/oauth/revoke HTTP/1.1
Host: www.hmislynk.com
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Accept: application/json
Accept-Charset: utf-8

access_token=ACCESS_TOKEN_GENERATED_BY_HMIS_LYNK
```

The HMISLynk authorization service indicates successful processing of the request by returning an HTTP status code 200 with the access token that has been revoked successfully.

An example response if the specified access token has been revoked successfully:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "access_token":"ACCESS_TOKEN_GENERATED_BY_HMIS_LYNK"
}
```

In future, additional parameters/fields may be included in the response. The client must be coded to ignore the parameters/fields it does not recognize.

For all error conditions, a status code 400 is used along with one of the following error responses.

Parameter Name	Description / Parameter Value	Required?
error	A single error code from the following: **invalid_request** The request is missing a required parameter, includes an unsupported parameter value, repeats a parameter, includes multiple tokens, or is otherwise malformed.	Yes

	server_error The HMISLynk authorization service encountered an unexpected condition which prevented it from fulfilling the request. **temporarily_unavailable** The HMISLynk authorization service is currently unable to handle the request due to a temporary overloading or maintenance of the server.	
error_description	A human-readable UTF-8 encoded text providing additional information, used to assist the application (aka client) developer in understanding the error that occurred.	No

An example error response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
"error":"ERROR_CODE_RETURNED_BY_HMIS_LYNK"
}
```