Infinity/Evolution Open API
OAuth 2.0 Specification

Version 1.5
August 15, 2016

Contents

1 OAuth 2.0 Overview

Authorization of access to the Infinity/Evolution Open API utilizes the OAuth 2.0 security authorization protocol, which was designed to grant limited client access to a resource. OAuth 2.0 is an Internet Engineering Task Force (IETF) standard defined by RFC 6749. Complete documentation on the standard can be found at http://tools.ietf.org/html/rfc6749.

Within the OAuth2 specification, there are four different authorization grant types that entail different overall authorization flows. “Authorization Code” and “Client Credential Grant” are the two supported methods for accessing the Infinity/Evolution Open API. The “Authorization Code” flow must be used when an individual user, who is registered in MyInfinity, is authorizing an application to access resources served by the Carrier Open API. The “Client Credential Grant” grant is a simplified authorization flow that should be used when a utility company application is accessing resources served by the Infinity/Evolution Open API and cannot be directly associated with an individual user registered in MyInfinity.

OAuth2 authorization is facilitated using the Hypertext Transfer Protocol, using Transport Layer Security (TLS) to encrypt the data connection (i.e. the HTTPS scheme). All interaction with the authorization server is via relatively simple REST web services. OAuth2 clients can be built by using HTTP GET and POST methods to access authorization URL’s. When accessing services with the HTTP GET method, parameters must be supplied as URL query parameters. When accessing services using the HTTP POST method, parameters must be supplied using the “application/x-www-form-urlencoded” format. The authorization server provides a numerical HTTP response code, as well as a message body in JavaScript Object Notation (JSON).

1.1 Key Terms

The OAuth2 specification uses the following key terms:

- **Resource Owner:** The owner of a resource to be accessed by an application. The owner could be the user accessing a system. For example, in the use case of an application that needs to retrieve the e-mail address and phone number of a user, the user might be considered the resource owner.
- **Client:** The application that authenticates and gains authorization with the authorization server, and subsequently uses this authorization to access a resource.
- **Authorization Server:** The server that authenticates and authorizes a client to access requested resources. Within this document, the authorization server is MyInfinity (https://www.myinfinitytouch.carrier.com).
- **Resource Server:** The server that provides a resource to clients, provided that the authorization to access the resources can be validated with the authorization server. Within this document, the resource server is the Open API (https://openapi.ing.carrier.com).
- **Authorization Endpoint:** The endpoint used by the client to obtain authorization from the resource owner via user-agent redirection. The client must make a GET request to the authorization endpoint by adding parameters to the query component of the authorization endpoint URI using the “application/x-www-form-urlencoded” format.
- **Token Endpoint:** The endpoint used by the client to exchange an authorization grant for an access token, typically with client authentication. The client must make a POST request to the token endpoint by sending the parameters using the “application/x-www-form-urlencoded” format with a character encoding of UTF-8.
- **Redirect Endpoint:** The client endpoint used by the authorization server to return responses containing authorization credentials to the client via the resource owner user-agent.
- **Authorization Code:** A code requested by a client from the authorization server. The code has a short lifespan and must be redeemed for an access token in order to gain authorization to a resource. Authorization codes are only used when an individual user that is registered in MyInfinityTouch is authorizing an application to access private resources on the Carrier Resource Server.
- **Access Token:** A tokenized string requested by a client from the authorization server. The token is passed to the resource server, which validates the token in order to grant access. The token has a limited lifespan, but can be refreshed using the refresh token.
- **Refresh Token:** A tokenized string provided to a client by the authorization server when an access token is requested. An access token can be renewed at

any time by using the refresh token. However, a refresh token cannot be used to access a resource.

- **Scope:** A space-delimited set of permissions that the application is requesting or an access token permits.
- **HTTP Basic Authentication:** A client must authenticate with the authorization server by providing its ID and “secret key” which is synonymous with a password. The client simply concatenates these two parameters, separated by a colon, Base64 encodes the concatenation and provides the result in the HTTP “Authorization” header. Example: Authorization: Basic tGzv3JOkF0XG5Qx2TlKWIA

2 Using OAuth 2.0 to Access the Infinity/Evolution Open API

All applications follow a basic pattern when accessing the Infinity/Evolution Open API using OAuth 2.0. Within this process there are four steps:

2.1 Obtain OAuth 2.0 Credentials From Carrier

In order to obtain OAuth 2.0 credentials, do the following:

1. Contact Carrier at InfinityOpenAPI@carrier.com, or EvolutionOpenAPI@Bryant.com and request a third-party application registration package.
2. Carrier will provide the complete Documentation Package for the Infinity/Evolution Open API, including an Application Questionnaire and the Infinity/Evolution Open API Third-Party Developer Agreement.
3. Return the completed Questionnaire and Developer Agreement to begin the approval process.
4. Upon approval, Carrier will provide you with samples of the Carrier Infinity Touch and Bryant Evolution Connex wall controls for your use in developing your application.
5. Carrier's partner, Outside Source Design (OSD), will provide a set of pre-production OAuth 2.0 credentials and URLs required for your client app.

2.2 Obtain an Access Token From the Carrier Authorization Server

Before your application can access private data using the Infinity/Evolution Open API, it must obtain an access token that grants access to that API. A variable parameter called "scope" controls the set of resources and operations that an access token permits. During the access-token request, your application sends one or more values in the "scope" parameter.

An authorization request requires an authentication step where the user logs in with their Carrier account. After logging in, the user is asked whether they are willing to grant the permissions that the application is requesting. This process is called *user consent*.

If the user grants the permission, the result of the sequence is an authorization code. The authorization code is returned as a query string parameter. The application exchanges the authorization code for an access token and a refresh token by making a request at token endpoint. During this exchange, the application authenticates with the authorization server using HTTP Basic authentication. If the user does not grant the permission, the server returns an error.

2.3 Send the Access Token to an API

After an application obtains an access token, it sends the token to a Carrier resource server in an HTTP authorization header. Access tokens are valid only for the set of operations and resources described in the “scope” of the token request. For example, if an access token is issued to view thermostats related information, it does not grant access to modify user information as well. You can, however, send that access token multiple times for permitted operations.

2.4 Refresh the Access Token (If Necessary)

Access tokens have limited lifetimes. If the application needs access to Carrier resource server beyond the lifetime of a single access token, it can obtain a refresh token. A refresh token allows the application to obtain new access tokens. A refresh token remains valid until the authorization is revoked explicitly. It is highly recommended that applications should save the refresh tokens in secure long-term storage and continue to use them as long as they remain valid or the authorization isn’t revoked.

3 “Authorization Code” Flow Message Specifications

Refer to the UML sequence diagram below depicting the “Authorization Code” message flow between client and Carrier authorization server. Detailed message specifications are found in subsequent sections.

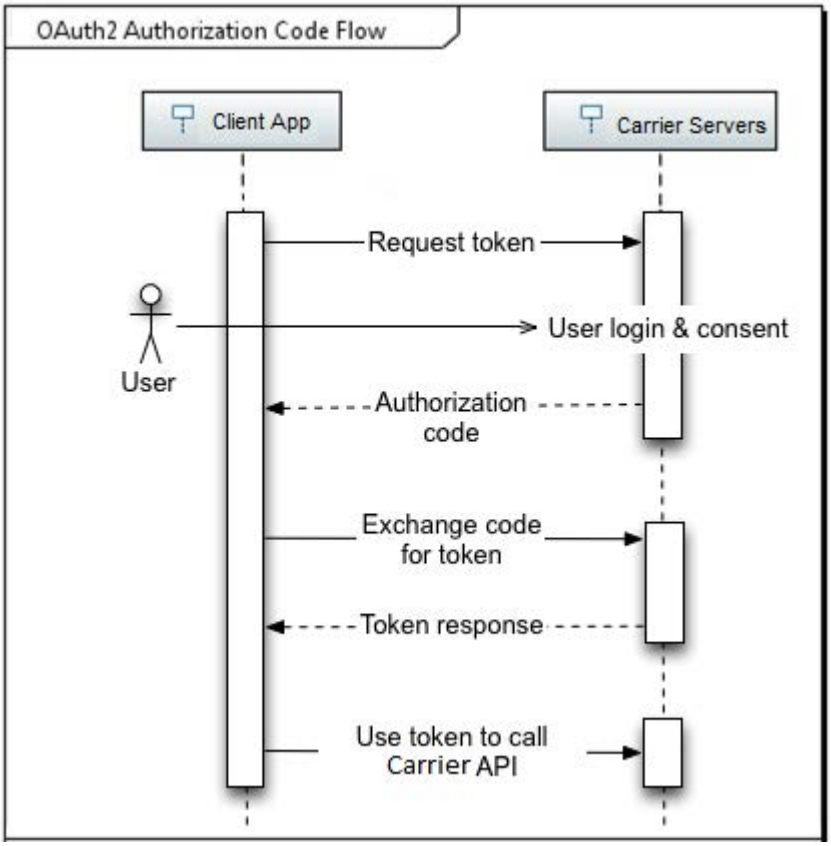

Figure 1 – “Authorization Code” Authentication & Authorization

3.1 Authentication to the Authorization Server

The “Auth” in OAuth stands for authorization, which is defined as a mechanism to control user or system access. In order to gain authorization to access the Infinity/Evolution Open API, a user or system client must first authenticate to the authorization server. With the “Authorization Code” grant type, an individual user must authenticate with the authorization server using a web browser. The OAuth client application controls the interaction between the web browser and the authorization server.

3.2 Authorization Request

The authentication message is an HTTPS GET message to the following URI:

https://www.myinfinitytouch.carrier.com/oauth2/authorize

The client constructs the request URI by adding the following parameters to the query component of the authorization endpoint URI using the “ application/x-www-form-urlencoded” format.

Parameter	Description
client_id	The client ID assigned to your application by Carrier.
redirect_uri	The response is sent to this URI and must match a URL previously established with the authorization server during the client registration process. This URI must use the HTTPS scheme for secure communication.
response_type	Value MUST be set to "code"
scope	A space-delimited set of permissions that the application is requesting.
state	An opaque value used by the client to maintain state between the request and callback. The authorization server includes this value when redirecting the user-agent back to the client. The parameter SHOULD be used for preventing cross-site request forgery. This parameter is optional but highly recommended.

Example:

```
GET https://www.myinfinitytouch.carrier.com/oauth2/authorize?
client_id=com.yourCompany.yourApp&
redirect_uri=yourApp%3A%2F%2FauthCode&
response_type=code&
scope=Read-System Write-System Read-User&
state=a1bcd2fgh3ijkLm4opQRST
```

Figure 2 – Authorization Request

The user will see the following two screens in succession:

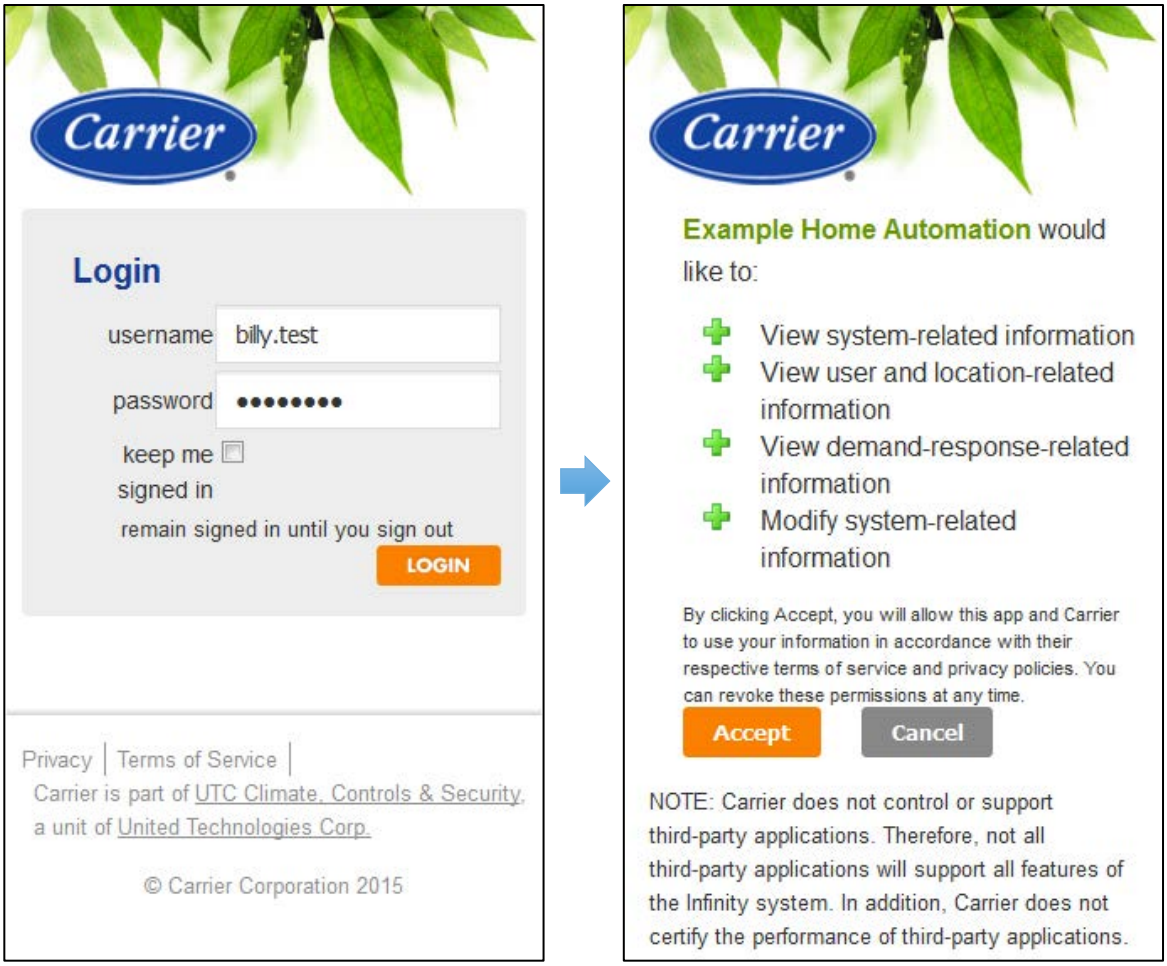

Figure 3 – Authorization and Authentication Screens Displayed to User

3.3 Available Scopes

Scope	Description
Read-System	View system-related information
Write-System	Modify system-related information
Read-User	View user and location-related information
Read-UtilityEvents	View demand-response-related information
Write-UtilityEvents	Modify demand-response-related information

3.4 Authorization Response

Responses are sent to the client with a 302 HTTP response code, which the browser interprets as a redirection command.

If the user grants the access request, the authorization server issues an authorization code and delivers it to the client by adding the following parameters to the query component of the redirection URI:

Parameter	Description
code	The authorization code generated by the Carrier authorization server. The authorization code expires shortly after it is issued to mitigate the risk of leaks.
state	If the "state" parameter was present in the client authorization request. The exact value received from the client.

Example:

```
HTTP/1.1 302 Found
Location: yourApp://authCode?
code=12A3456BCD789123&
state=a1bcd2fgh3ijkLm4opQRST
```

Figure 4 – Authorization Response

3.5 Authorization Error Response

If the request fails due to a missing, invalid, or mismatching redirection URI, or if the client identifier is missing or invalid, the authorization server will inform the user of the error and does not automatically redirect the user-agent to the 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 authorization server informs the client by adding the following parameters to the query component of the redirection URI:

Parameter	Description
error	The error code generated by the Carrier authorization server.
error_description	The description of the authorization error to assist the client developer in understanding the error that occurred. This is optional parameter.
error_uri	A URI identifying a human-readable web page with information about the error, used to provide the client developer with additional information about the error. This is optional parameter.
state	If the "state" parameter was present in the client authorization request. The exact value received from the client.

Error Example:

```
HTTP/1.1 302 Found
Location: yourApp://authCode?
error=access_denied&
error_description=User%rejected%20the%20request&
state=a1bcd2fgh3ijkLm4opQRST
```

Figure 5 – Authorization Error Response

3.6 Access Token Request

The request access token message is an HTTPS POST message to the following URI:

https://www.myinfinitytouch.carrier.com/oauth2/token

The request must include the following headers:

Header	Description
Authorization	Basic HTTP authentication string for client
Content-Type	Value MUST be set to “application/x-www-form-urlencoded”

The client makes a request to the token endpoint by sending the following parameters using the “application/x-www-form-urlencoded” format with a character encoding of UTF-8.

Parameter	Description
code	The authorization code generated by the Carrier authorization server.
grant_type	Value MUST be set to “authorization_code”
redirect_uri	The value sent in the previous authorization request. This must match a URL previously established with the authorization server during the client registration process.

Example:

```
POST /oauth2/token HTTP/1.1
Host: www.myinfinitytouch.carrier.com
Authorization: Basic tGzv3JOkF0XG5Qx2TlKWIA
Content-Type: application/x-www-form-urlencoded

code=12A3456BCD789123&
grant_type=authorization_code&
redirect_uri= yourApp%3A%2F%2FauthCode
```

Figure 6 – Access Token Request

3.7 Access Token Response

The access token response is a synchronous HTTPS response to the HTTPS POST message sent by the client. If the access token request is valid and authorized, the authorization server issues an access token and refresh token. If the request client authentication failed or is invalid, the authorization server returns an error response. Authorization server also specifies the authentication scheme that should be used inside “WWW-Authenticate” header when returning an unauthorized response.

The response is formatted in JavaScript Object Notation (JSON), with the following parameters:

Parameter	Description
access_token	The access token generate by the Carrier authorization server.
expires_in	The remaining validity of the access token, measured in seconds.
token_type	The type of the access token.
refresh_token	A token that can be used to obtain a new access token.
scope	A space-delimited set of permissions that access token permits.

Example:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache

{
"access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":"bearer",
"expires_in":3600,
"refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
"scope": "Read-System Write-System"
}
```

Figure 7 – Access Token Response

3.8 Access Token Error Response

The access token request error response is formatted in JavaScript Object Notation (JSON), with the following parameters:

Parameter	Description
error	The error code generated by the Carrier authorization server.
error_description	The description of the authorization error to assist the client developer in understanding the error that occurred. This is optional parameter.
error_uri	A URI identifying a human-readable web page with information about the error, used to provide the client developer with additional information about the error. This is optional parameter.

Error Example:

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json; charset=utf-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache

{
"error":"invalid_client"
"error_description":"The client secret was incorrect"
}
```

Figure 8 – Access Token Error Response Example

4 “Client Credential” Flow Message Specifications

Refer to the UML sequence diagram below depicting the “Client Credential” message flow between client and Carrier authorization server. Detailed message specifications are found in subsequent sections.

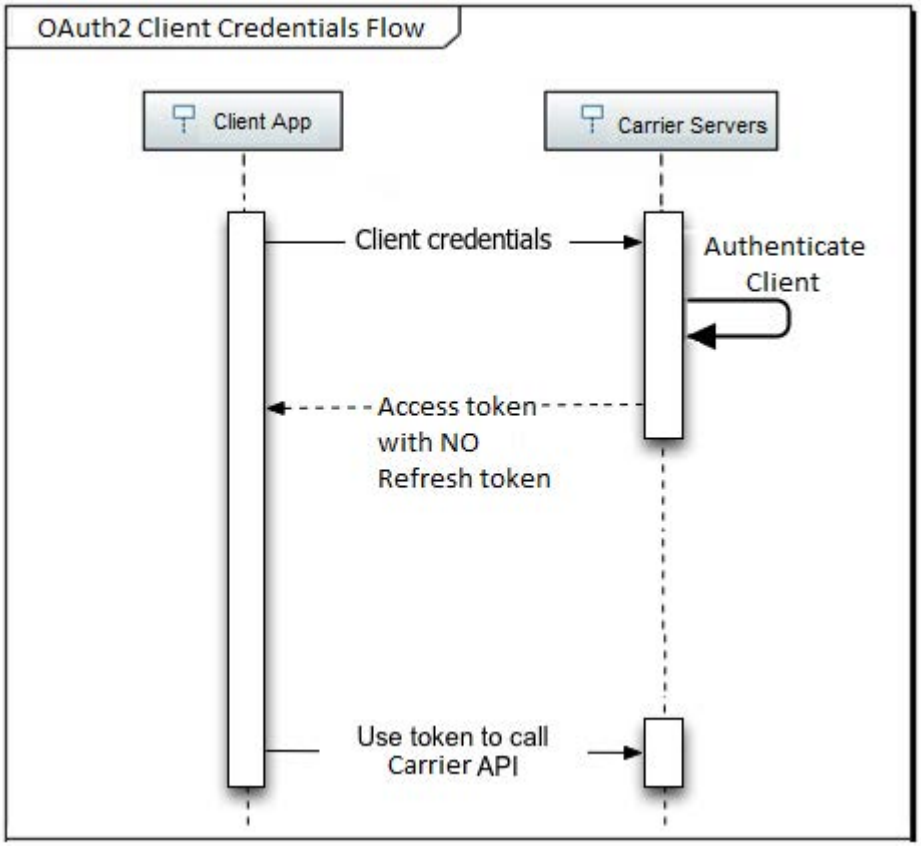

Figure 9 – “Client Credentials” Authentication & Authorization

4.1 Authentication to the Authorization Server

The “Auth” in OAuth stands for authorization, which is defined as a mechanism to control user or system access. In order to gain authorization to access the Infinity/Evolution Open API, a user or utility client must first authenticate to the authorization server. With the “Client Credential” grant type, the Client is acting on its own behalf (the client is also the resource owner) or is requesting access to protected resources based on an authorization previously arranged with the authorization server. The client authenticates with the authorization server by providing its ID and “secret key” which is synonymous with a password. The client simply concatenates these two parameters, separated by a colon, Base64 encodes the concatenation and provides the result in the HTTP “Authorization” header. Only **confidential** clients get an access token using this flow.

Example:

Authorization: Basic tGzv3JOkF0XG5Qx2TlKWIA

4.2 Available Scopes

Scope	Description
Read-System	View system-related information
Write-System	Modify system-related information
Read-User	View user and location-related information
Read-UtilityEvents	View demand-response-related information
Write-UtilityEvents	Modify demand-response-related information

4.3 Access Token Request

The request access token message is an HTTPS POST message to the following URI:

https://www.myinfinitytouch.carrier.com/oauth2/token

The request must include the following headers:

Header	**Description**
Authorization	Basic HTTP authentication string for client
Content-Type	Value MUST be set to “application/x-www-form-urlencoded”

The client makes a request to the token endpoint by sending the following parameters using the “application/x-www-form-urlencoded” format with a character encoding of UTF-8:

Parameter	Description
grant_type	Value MUST be set to “client_credentials”
scope	A space-delimited set of permissions that the application is requesting. This is an optional parameter. If the “scope” parameter is not present, the access token will be issued without any permissions in order to gain access to only limited APIs on the resource server.

Example:

```
POST /oauth2/token HTTP/1.1
Host: www.myinfinitytouch.carrier.com
Authorization: Basic tGzv3JOkF0XG5Qx2TlKWIA
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
```

Figure 10 – Access Token Request

4.4 Access Token Response

The access token response is a synchronous HTTPS response to the HTTPS POST message sent by the client. If the access token request is valid and authorized, the authorization server issues an access token and refresh token. If the request client authentication failed or is invalid, the authorization server returns an error response. The authorization server also specifies the authentication scheme that should be used inside “WWW-Authenticate” header when returning an unauthorized response.

The response is formatted in JavaScript Object Notation (JSON), with the following parameters:

Parameter	Description
access_token	The access token generate by the Carrier authorization server.
expires_in	The remaining validity of the access token, measured in seconds.
token_type	The type of the access token.
scope	A space-delimited set of permissions that access token permits. Returned only if the "scope" parameter was present in the client authorization request.

Example:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache

{
"access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":"bearer",
"expires_in":3600,
"scope": "Read-System Write-System Write-UtilityEvents"
}
```

Figure 11 – Access Token Response

4.5 Access Token Error Response

The access token request error response is formatted in JavaScript Object Notation (JSON), with the following parameters:

Parameter	Description
error	The error code generated by the Carrier authorization server.
error_description	The description of the authorization error to assist the client developer in understanding the error that occurred. This is optional parameter.
error_uri	A URI identifying a human-readable web page with information about the error, used to provide the client developer with additional information about the error. This is optional parameter.

Error Example:

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json; charset=utf-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache

{
"error":"invalid_client",
"error_description":"The client secret was incorrect"
}
```

Figure 12 – Access Token Error Response

5 Invoking a Business Service

Business services include REST web services accessible via one of the HTTP methods - GET, POST, PUT or DELETE. The Carrier resource server provides a numerical HTTP response code, as well as a message body in JavaScript Object Notation (JSON) or XML format based on accept headers set by the client application while making the request accessing Carrier resource server APIs. When protected via OAuth 2.0, accessing the service requires specifying the “Authorization” HTTP header with the value equal to the access token type (i.e. “Bearer”) followed by the access token as shown below:

Business Service Invocation Example:

```
GET /user HTTP/1.1
Authorization: Bearer tGzv3JOkF0XG5Qx2TlKWIA
Host: openapi.ing.carrier.com
```

Figure – Business Service Invocation Example

The resource server also specifies the authentication scheme that should be used inside “WWW-Authenticate” header when returning an unauthorized response.

6 Refreshing Access Tokens

A refresh token can be used to refresh a short-lived access token. The authorization server returns an access token along with a new refresh token for every successful refresh access token request. A refresh token remains valid as long as the authorization hasn't been revoked explicitly and it has been used at least once in the last six months. The authorization server also specifies the authentication scheme that should be used inside "WWW-Authenticate" header when returning an unauthorized response.

6.1 Refresh Access Token Request

The refresh access token request is an HTTPS POST message to the following,

https://www.myinfinitytouch.carrier.com/oauth2/token

The request must include the following headers:

Header	Description
Authorization	Basic HTTP authentication string for client.
Content-Type	Value MUST be set to "application/x-www-form-urlencoded"

The client makes a request to the token endpoint by sending the following parameters using the "application/x-www-form-urlencoded" format with a character encoding of UTF-8:

Parameter	Description
grant_type	Value MUST be set to "refresh_roken"
refresh_token	The refresh token granted during initial authorization. A refresh token remains valid as long as the authorization hasn't been revoked explicitly and it has been used at least once in last six months.

Example:

```
POST /oauth2/token HTTP/1.1
Host: www.myinfinitytouch.carrier.com
Authorization: Basic tGzv3JOkF0XG5Qx2TlKWIA
Content-Type: application/x-www-form-urlencoded

grant_type= refresh_token&
refresh_token=fdisofjdof!fkdlqwe343lT
```

Figure 13 – Refresh Access Token Request

6.2 Refresh Access Token Response

The refresh access token response is a synchronous HTTPS response to the HTTPS POST message sent by the client. The message body is formatted in JavaScript Object Notation (JSON), with the following parameters:

Parameter	Description
access_token	The access token generate by the Carrier authorization server.
scope	A space-delimited set of permissions that access token permits.
expires_in	The remaining validity of the access token, measured in seconds.
token_type	The type of the access token
refresh_token	The new refresh token to be used for subsequent refresh.

Example:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache

{
"access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":"bearer",
"expires_in":3600,
"refresh_token":"823BHjfFHAAA12345OLks",
"scope":"Read-System Write-System"
}
```

Figure 14 – Refresh Access Token Response

6.3 Refresh Access Token Error Response

The refresh access token request error response is formatted in JavaScript Object Notation (JSON), with the following parameters:

Parameter	Description
error	The error code generated by the Carrier authorization server.
error_description	The description of the authorization error to assist the client developer in understanding the error that occurred. This is optional parameter.
error_uri	A URI identifying a human-readable web page with information about the error, used to provide the client developer with additional information about the error. This is optional parameter.

Error Example:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache

{
"error":"invalid_grant"
}
```

Figure 15 – Refresh Access Token Error Response