Destiny Open API's

(Developer's Guide)
Version 1.0

1. Introduction

Note: As of Destiny 16.5, the Destiny Open API's are in beta.

The Destiny Open API's provide a way for application developers to interact with the Destiny application server using a series of RESTful web services.

The goal of this document is to help build familiarity with the Destiny Open API's from a development standpoint in order to simplify the integration process.

Two main sections are presented in this guide. The first section (About Destiny Open API's) gives an overview of the Destiny Open API architecture as well as patterns and rules of thumb to follow when developing API clients. The second section (Getting Started) walks you through setting up an access token and testing it against the status service.

Once you can connect to the status service, you are ready to start working with an API endpoint and should refer to the API endpoint specification. This developer guide will not go into the details of each API endpoint.

All Destiny Open API endpoints are documented and can be viewed in the embedded Swagger UI running on the Destiny server. If you know the Destiny host, you should be able to access Destiny Open API's specifications using the following paths.

Authentication -- https://<destiny_host>/api/v1/doc/auth-api.jsp
Status -- https://<destiny_host>/api/v1/doc/status-api.jsp
Fines - https:// <destiny_host>/api/v1/doc/fines-api.jsp

2. About Destiny Open API's

This section helps to build an understanding of how Destiny Open API's are architected as well as provide rules that clients will need to follow.

2.1. URL Path Structure

A Destiny Open API endpoint is broken down into the following URL path segments

Template:
https://<host>/api/<version>/rest/context/<context>/sites/<site>/<endpoint>

Example:
https://localhost.edu/api/v1/rest/context/miami_dade/sites/100/patrons/284

- **<host>** – required
  The fully qualified domain name of the Destiny server.
- **<version>** – required
  The major API version of the endpoint. More on this later
- **/context/<context>** – optional
  Context name of the Destiny school district, when the Destiny server is hosting multiple districts. The Destiny administrator should be able to help determine what the context name is for a given district. This path segment can be left out when Destiny is a single district server. Or you can alternatively use "/context/destiny" in that case.
- **/sites/<site>** – optional
  Destiny site to limit access to. Whether or not this site path segment is supported varies by API endpoint. The API specification for the given endpoint will tell you if you can use it or not. It can be left off the URL path when not needed.
- **<endpoint>** – required
  This is the last part of the URL path, which should represent the resource being accessed.

2.2. Data exchanged in JSON format

Destiny exchanges resource data using the JSON format. The appropriate content type header will be set when a response contains resource data.

Content-Type: application/json

Destiny does not currently have an option for the client to send or receive information in XML format.

2.3. Version Management

The Destiny Open API's are version managed using a <major>.<minor> pattern.

Minor changes to API's are considered to be non-breaking. This means that a client should not need to make any code changes or experience any negative side effects from migrating to a newer minor version of the API. The following examples would be considered minor changes:

- Adding a new field or data structure to an existing JSON object
- Adding new API services or service endpoints
- Adding optional query parameters to an existing API endpoint

Major changes to the API are considered to be breaking. This means that the client will likely be required to make code changes to ensure compatibility when migrating to the newer version. The following are examples of major (breaking) changes:

- Removing an existing API endpoint
- Renaming existing JSON fields being sent or received by an existing endpoint
- Changing HTTP method of an existing endpoint
- Changing HTTP status codes of an existing endpoint
- Changing HTTP resource path of an existing endpoint
- Changing data types or data structure of JSON

As a consequence of the impacts of a breaking change, the major version is included in the path of all Destiny Open API endpoints.

Example: https://localhost/api/v1/rest/service/endpoint

Destiny will be able to host multiple major versions of the Destiny Open API's simultaneously, each to a different version path of the URL. The goal is to allow clients time to make code changes needed to support a major migration. Older versions of the Destiny Open API's will have a limited window of ongoing support before they are obsoleted.

Some development frameworks can be very strict in their default settings. To ensure that minor version updates do not break clients, it is important to allow flexibility when exchanging JSON information between clients. Many JSON fields do not need to be strictly required or absent to allow a successful API request. Clients should try to adhere to the following rules of thumb to ensure resilience to non-breaking changes.

- Allow for JSON fields to be sent or received which are not documented or expected. They can be ignored if not understood.
- Allow for JSON fields which are present, but not required, in documentation to be missing, null or empty.

2.4. Error Handling

When Destiny Open API requests are successful, status codes in the 2xx and 3xx range will be given. Any status code in the 4xx or 5xx range is considered to be an error or a fault and will have special error handling.

When Destiny encounters an error during the processing of an API request, a response will be sent with an appropriate HTTP status code and a JSON error object containing information about the error. Please note that the status code is part of the HTTP message and not contained in the error object.

This is an example of the error object

```
{
  "error" : {
    "code" : "INVALID_FIELD",
    "target": "patronBarcode",
    "message" : "Patron barcode \"V 10101\" is invalid.",
    "logId" : "1aF3v"
  }
}
```

- **code** – A finite list of machine friendly codes describing the type of error. Error codes can vary by API endpoint. The API specification for each endpoint should document what codes could be returned.
- **target** – Used to specify either a JSON field or URL resource path which is associated with the error.
- **message** – A human readable error message, intended for the consumption of the client developer or support person. These messages are not formatted for end user consumption.
- **logId** – Will be included when something useful regarding this error may have been written to the Destiny log files. The value is a random five character string, which should be unique in the log, thus making searches more effective.

Following are HTTP status codes that may be returned:

- **400 (BAD REQUEST)** – The client request was malformed.
- **401 (UNAUTHORIZED)** - The request requires client authentication or the client credentials were invalid.
- **403 (FORBIDDEN)** - Server refuses to fulfill request for resource.

- **404 (NOT FOUND)** - No resource matches the given request URL.
- **405 (METHOD NOT ALLOWED)** - The wrong HTTP method was used in making the request.
- **500 (INTERNAL SERVER ERROR) –** An unanticipated issue occurred while the request was being processed. The logId of the error object becomes particularly important when 500 errors occur.

2.5. Access Tokens

Note: The "Getting Started" section of this guide will show more detail about obtaining and using an access token.

Destiny Open API endpoints are secured with OAuth2 access tokens. When an API account is setup, by the Destiny administrator, a Client ID and Client Secret are created. These random alpha-numeric strings are the credentials for obtaining an access token. The credentials should be shared discretely and securely between the Destiny administrator and the client developer/configurator and must be protected from access by unwanted parties. It is wise to treat the credentials confidentially, like you would any other username/password sequence.

Example Credentials:

Client ID: xApkF3cE49wM2G-p3CvZ40fxCQUszrM=
Client Secret: Ch_9qbOrlJhbHxF8eSJq4PRvzE+qWA8=

To create an access token, the client must make an API call to the Destiny OAuth2 access token API. See the "Getting Started" section.

Example Access Token:

```
{
 "access_token" : "5nIAfwTX1ZebNVYFjp2iFFG7RHfmsPmrQIbTylHOZgc",
 "token_type" : "Bearer",
 "expires_in" : 3600
}
```

Once an access token is created it can be used to make subsequent API calls. The access token is created with URL safe alpha-numeric characters, meaning that no special action needs to be taken to escape them. When making an API call, the access token can be provided in one of two ways.

Authorization Header:
Authorization: Bearer 5nIAfwTX1ZebNVYFjp2iFFG7RHfmsPmrQIbTylHOZgc

Query Parameter:
http://localhost/api/v1/rest/context/destiny/fines?accessToken=5nIAfwTX1ZebNVYFjp2iFFG7RHfmsPmrQIbTylHOZgc

However, use of the query parameter is strongly discouraged as it can create undesired side effects. For example, HTTP caching would be impacted by expiration of the token and a new one being created.

Destiny access tokens have a finite time to live, expressed as "expires_in" in the token creation response. The value represents the number of seconds that the token will live after it was created. The client can use this information to anticipate the need to create another access token before the existing one expires.

Destiny currently does not use refresh tokens.

Any attempt to access a Destiny Open API's with an expired token will result in a 401 HTTP status code and error code (CODE_INVALID_TOKEN) indicating that the token is invalid. The client can optionally use this response to detect an expired access token, create a new one, and then retry the API call.

The Destiny Administrator can revoke API account credentials by deleting the associated API credential associated with them. Subsequent attempts to obtain an access token with the same credentials will be refused.

2.6. Use Secure Connections

While Destiny Open API's may accept connections over HTTP, it is imperative that the client use a secure HTTPS connection when making API calls to Destiny. Please ensure that the following guidelines are followed when connections are established.

- Limit protocol to TLS v1.1 or better
- Use strong cipher suites
- Validate the certificate path
- Validate the certificate host name matches host of the URL

Using HTTPS connections are absolutely required to ensure that information exchanged between Destiny and the consumer application is safeguarded against snooping from

unwanted parties. Data exchanged with Destiny Open API's can contain information which is personally identifiable in nature and should remain private.

2.7. Caching

While no specific Destiny Open API endpoint relies on caching yet, Destiny will manage caching where appropriate using the standard cache control headers. Ensure that the client frameworks used recognize and respect cache control headers.

3. Getting Started

Note: During API client development and testing, please refrain from accessing any production Destiny servers. Appropriate test servers with sanitized test data must be used. Please contact Follett School Solutions about test servers that you may utilize during the early stages of your development process.

Before you begin with this section, it is important that the Destiny administrator has set up an API account and has given you the client ID and client secret.

For the purpose of simplicity, the local loopback ("localhost") will be used in place of a fully qualified domain name in the URL examples. Please use the appropriate host when testing.

Note: Destiny servers can be installed with either a single tenant (district) or multiple tenants (consortium/distributed). The examples here treat Destiny as a single tenant (/api/v1/rest/context/destiny). Don't forget to use context name of the tenant you intend to communicate with when accessing a multi-tenant Destiny server. Please refer to the"URL Path Structure" section of this document and consult the Destiny Administrator to find out what the context name is for the intended district.

3.1. Obtain an Access Token

Before making any Destiny Open API calls, an access token must first be acquired. Destiny exposes an OAuth2 access token endpoint in which a "client_credentials" grant_type can be used to obtain a bearer token. This API accepts the token request in a form post body and will return a JSON object containing the bearer token. Please note that the parameters of the post body are not accepted as query parameters.

Request

```
POST https://localhost/api/v1/rest/context/destiny/auth/accessToken HTTP/1.1
Host: localhost
User-Agent: Java/1.8.0_183
Content-Type: application/x-www-form-urlencoded
Content-Length: 124

?grant_type=client_credentials&client_id=QTwFr9UhANkaFbPksOaVk7g3dU4Zc5p%
3D&client_secret=zln+9XGrdVlFBaMUlPM0RcOqhofqmQ5%3D
```

Form Parameters:

grant_type – Set this to “client_credentials”
client_id – Set this to the Client ID given to you by the Destiny administrator
client_secret – Set this to the Client Secret given to you by the Destiny administrator

Response

```
HTTP/1.1 200 OK
Date: Mon, 17 Dec 2018 21:41:28 GMT
Server: Destiny
X-Powered-By: Follett School Solutions
Content-Type: application/json; charset=utf-8
Content-Length: 119

{
 "access_token" : "5nIAfwTX1ZebNVYFjp2iFFG7RHfmsPmrQIbTylHOZgc",
 "token_type" : "Bearer",
 "expires_in" : 3600
}
```

JSON Fields

access_token – The access token used to make subsequent API calls.
token_type – Always set to “Bearer” which will also be expected in the authorization header.
expires_in – The number of seconds the access token will be valid for as computed from the moment of request.

3.2. Make an API Request

Now we can make an API request, placing the access token in the authorization header.

Authorization: Bearer<space><access_token>

Example:
Authorization: Bearer 5nIAfwTX1ZebNVYFjp2iFFG7RHfmsPmrQIbTylHOZgc

The status service is a very simple API, and a great way to test the access token. As long as the access token is valid and passed into the API correctly, the request should succeed and a very simple JSON object with version information about Destiny Open API's will be returned.

Request

```
GET https://localhost/api/v1/rest/context/destiny/status HTTP/1.1
Host: localhost
User-Agent: Java/1.8.0_183
Authorization: Bearer 5nIAfwTX1ZebNVYFjp2iFFG7RHfmsPmrQIbTylHOZgc
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 17 Dec 2018 21:41:28 GMT
Server: Destiny
X-Powered-By: Follett School Solutions
Content-Type: application/json; charset=utf-8
Content-Length: 75

{
 "fullVersion" : "1.0",
 "majorVersion" : 1,
 "minorVersion" : 0
}
```

JSON Fields

fullVersion – The full version of the Destiny Open API in string format.
majorVersion – The major version of the Destiny Open API in integer format. It will be the same version as in the URL.
minorVersion – The minor version of the Destiny Open API in integer format. Useful for detecting if Destiny Open API's will support features added since a given minor version.

3.3. Ready to Go

At this point, if your client is able to successfully create an access token and invoke the status service, you are ready to make real API calls. Consult the appropriate API specification for the feature of interest to proceed.