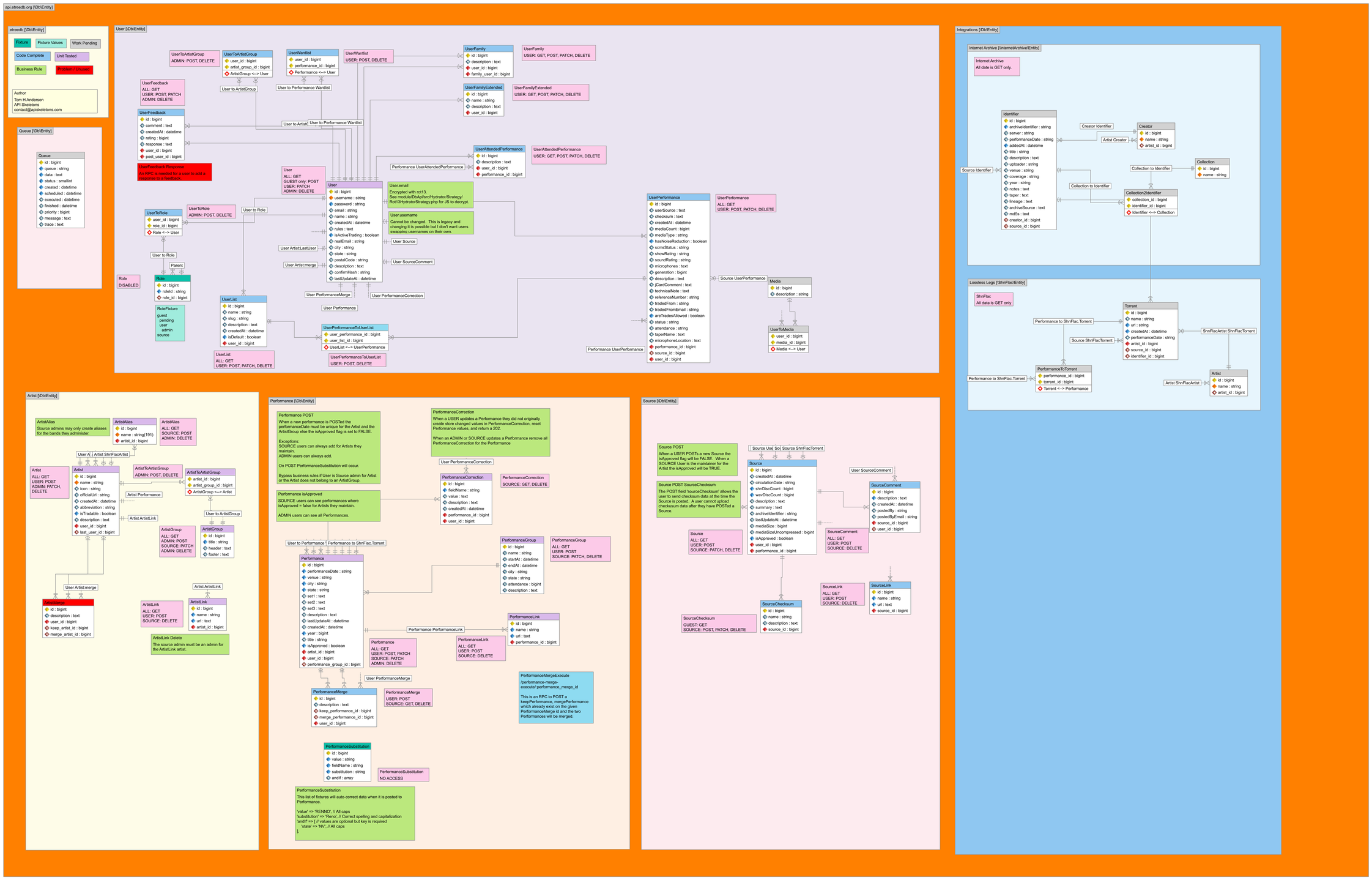
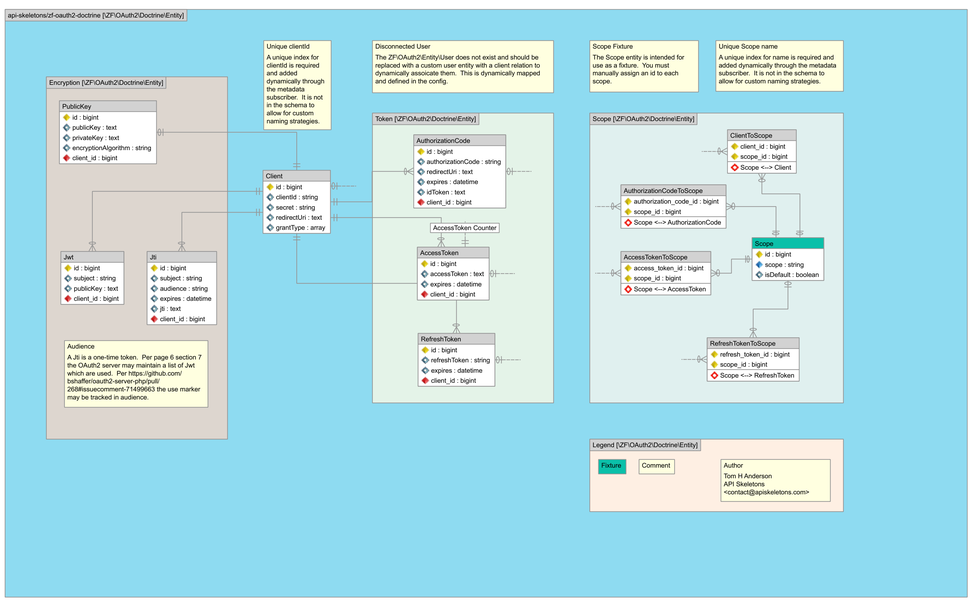
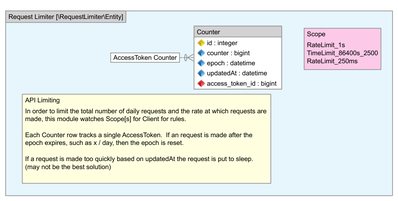

api.etreedb.org [/Db/Entity]
etreedb [/Db/Entity]
Fixture
Fixture Values
Work Pending
Code Complete
Unit Tested
Business Rule
Problem / Unused
Author
Tom H Anderson
API Skeletons
contact@apiskeletons.com
Queue [/Db/Entity]
Queue
id : bigint
queue : string
data : text
status : smallint
created : datetime
scheduled : datetime
executed : datetime
finished : datetime
priority : bigint
message : text
trace : text
User [/Db/Entity]
UserToArtistGroup
ADMIN: POST, DELETE
UserToArtistGroup
user_id : bigint
artist_group_id : bigint
ArtistGroup <--> User
User to ArtistGroup
UserWantlist
user_id : bigint
performance_id : bigint
Performance <--> User
User to Performance Wantlist
UserWantlist
USER: POST, DELETE
UserFamily
id : bigint
description : text
user_id : bigint
family_user_id : bigint
UserFamily
USER: GET, POST, PATCH, DELETE
UserFamilyExtended
id : bigint
name : string
description : text
user_id : bigint
UserFamilyExtended
USER: GET, POST, PATCH, DELETE
UserFeedback
ALL: GET
USER: POST, PATCH
ADMIN: DELETE
UserFeedback
id : bigint
comment : text
createdAt : datetime
rating : bigint
response : text
user_id : bigint
post_user_id : bigint
User to Artist
User to Performance Wantlist
UserFeedback Response
An RPC is needed for a user to add a response to a feedback.
UserAttendedPerformance
id : bigint
description : text
user_id : bigint
performance_id : bigint
UserAttendedPerformance
USER: GET, POST, PATCH, DELETE
Performance UserAttendedPerformance
User
ALL: GET
GUEST only: POST
USER: PATCH
ADMIN: DELETE
User
id : bigint
username : string
password : string
email : string
name : string
createdAt : datetime
rules : text
isActiveTrading : boolean
realEmail : string
city : string
state : string
postalCode : string
description : text
confirmHash : string
lastUpdateAt : datetime
User.email
Encrypted with rot13.
See module/DbApi/src/Hydrator/Strategy/Rot13HydratorStrategy.php for JS to decrypt.
User.username
Cannot be changed. This is legacy and changing it is possible but I don't want users swapping usernames on their own.
User Source
User SourceComment
UserToRole
user_id : bigint
role_id : bigint
Role <--> User
UserToRole
ADMIN: POST, DELETE
User to Role
User to Role
Parent
Role
DISABLED
Role
id : bigint
roleId : string
role_id : bigint
RoleFixture
guest
pending
user
admin
source
User Artist:LastUser
User Artist:merge
User PerformanceMerge
User PerformanceCorrection
User Performance
UserList
id : bigint
name : string
slug : string
description : text
createdAt : datetime
isDefault : boolean
user_id : bigint
UserList
ALL: GET
USER: POST, PATCH, DELETE
UserPerformanceToUserList
user_performance_id : bigint
user_list_id : bigint
UserList <--> UserPerformance
UserPerformanceToUserList
USER: POST, DELETE
UserPerformance
id : bigint
userSource : text
checksum : text
createdAt : datetime
mediaCount : bigint
mediaType : string
hasNoiseReduction : boolean
scmsStatus : string
showRating : string
soundRating : string
microphones : text
generation : bigint
description : text
jCardComment : text
technicalNote : text
referenceNumber : string
tradedFrom : string
tradedFromEmail : string
areTradesAllowed : boolean
status : string
attendance : string
taperName : text
microphoneLocation : text
performance_id : bigint
source_id : bigint
user_id : bigint
UserPerformance
ALL: GET
USER: POST, PATCH, DELETE
Source UserPerformance
Media
id : bigint
description : string
UserToMedia
user_id : bigint
media_id : bigint
Media <--> User
Performance UserPerformance
Integrations [/Db/Entity]
Internet Archive [/InternetArchive/Entity]
Internet Archive
All data is GET only.
Identifier
id : bigint
archiveIdentifier : string
server : string
performanceDate : string
addedAt : datetime
title : string
description : text
uploader : string
venue : string
coverage : string
year : string
notes : text
taper : text
lineage : text
archiveSource : text
md5s : text
creator_id : bigint
source_id : bigint
Creator Identifier
Artist Creator
Creator
id : bigint
name : string
artist_id : bigint
Source Identifier
Collection to Identifier
Collection to Identifier
Collection
id : bigint
name : string
Collection2Identifier
collection_id : bigint
identifier_id : bigint
Identifier <--> Collection
Lossless Legs [/ShnFlac/Entity]
ShnFlac
All data is GET only
Performance to ShnFlac.Torrent
Torrent
id : bigint
name : string
url : string
createdAt : datetime
performanceDate : string
artist_id : bigint
source_id : bigint
identifier_id : bigint
ShnFlacArtist ShnFlacTorrent
Source ShnFlacTorrent
PerformanceToTorrent
performance_id : bigint
torrent_id : bigint
Torrent <--> Performance
Performance to ShnFlac.Torrent
Artist ShnFlacArtist
Artist
id : bigint
name : string
artist_id : bigint
Artist [/Db/Entity]
ArtistAlias
Source admins may only create aliases for the bands they administer.
ArtistAlias
id : bigint
name : string(191)
artist_id : bigint
ArtistAlias
ALL: GET
SOURCE: POST
ADMIN: DELETE
User A
Artist ShnFlacArtist
Artist
ALL: GET
USER: POST
ADMIN: PATCH, DELETE
Artist
id : bigint
name : string
icon : string
officialUrl : string
createdAt : datetime
abbreviation : string
isTradable : boolean
description : text
user_id : bigint
last_user_id : bigint
ArtistToArtistGroup
ADMIN: POST, DELETE
ArtistToArtistGroup
artist_id : bigint
artist_group_id : bigint
ArtistGroup <--> Artist
Artist Performance
Artist ArtistLink
User to ArtistGroup
ArtistGroup
ALL: GET
ADMIN: POST
SOURCE: PATCH
ADMIN: DELETE
ArtistGroup
id : bigint
title : string
header : text
footer : text
User Artist:merge
Artist ArtistLink
ArtistMerge
id : bigint
description : text
user_id : bigint
keep_artist_id : bigint
merge_artist_id : bigint
ArtistLink
ALL: GET
USER: POST
SOURCE: DELETE
ArtistLink
id : bigint
name : string
url : text
artist_id : bigint
ArtistLink Delete
The source admin must be an admin for the ArtistLink artist.
Performance [/Db/Entity]
Performance POST
When a new performance is POSTed the performanceDate must be unique for the Artist and the ArtistGroup else the isApproved flag is set to FALSE.
Exceptions:
SOURCE users can always add for Artists they maintain.
ADMIN users can always add.
On POST PerformanceSubstitution will occur.
Bypass business rules if User is Source admin for Artist or the Artist does not belong to an ArtistGroup.
PerformanceCorrection
When a USER updates a Performance they did not originally create store changed values in PerformanceCorrection, reset Performance values, and return a 202.
When an ADMIN or SOURCE updates a Performance remove all PerformanceCorrection for the Performance
User PerformanceCorrection
PerformanceCorrection
id : bigint
fieldName : string
value : text
description : text
createdAt : datetime
performance_id : bigint
user_id : bigint
PerformanceCorrection
SOURCE: GET, DELETE
Performance isApproved
SOURCE users can see performances where isApproved = false for Artists they maintain.
ADMIN users can see all Performances.
User to Performance
Performance to ShnFlac.Torrent
Performance
id : bigint
performanceDate : string
venue : string
city : string
state : string
set1 : text
set2 : text
set3 : text
description : text
lastUpdateAt : datetime
createdAt : datetime
year : bigint
title : string
isApproved : boolean
artist_id : bigint
user_id : bigint
performance_group_id : bigint
PerformanceGroup
id : bigint
name : string
startAt : datetime
endAt : datetime
city : string
state : string
attendance : bigint
description : text
PerformanceGroup
ALL: GET
USER: POST
SOURCE: PATCH, DELETE
Performance PerformanceLink
PerformanceLink
id : bigint
name : string
url : text
performance_id : bigint
Performance
ALL: GET
USER: POST, PATCH
SOURCE: PATCH
ADMIN: DELETE
PerformanceLink
ALL: GET
USER: POST
SOURCE: DELETE
User PerformanceMerge
PerformanceMerge
id : bigint
description : text
keep_performance_id : bigint
merge_performance_id : bigint
user_id : bigint
PerformanceMerge
USER: POST
SOURCE: GET, DELETE
PerformanceMergeExecute
/performance-merge-execute/:performance_merge_id
This is an RPC to POST a keepPerformance, mergePerformance which already exist on the given PerformanceMerge id and the two Performances will be merged.
PerformanceSubstitution
id : bigint
value : string
fieldName : string
substitution : string
andIf : array
PerformanceSubstitution
NO ACCESS
PerformanceSubstitution
This list of fixtures will auto-correct data when it is posted to Performance.
'value' => 'RENNO', // All caps
'substitution' => 'Reno', // Correct spelling and capitalization
'andIf' => [// values are optional but key is required
'state' => 'NV', // All caps
],
Source [/Db/Entity]
Source POST
When a USER POSTs a new Source the isApproved flag will be FALSE. When a SOURCE User is the maintainer for the Artist the isApproved will be TRUE.
Source POST SourceChecksum
The POST field 'sourceChecksum' allows the user to send checksum data at the time the Source is posted. A user cannot upload checksum data after they have POSTed a Source.
Source Use
Source ShnFlacTorrent
Source
id : bigint
createdAt : datetime
circulationDate : string
shnDiscCount : bigint
wavDiscCount : bigint
description : text
summary : text
archiveIdentifier : string
lastUpdateAt : datetime
mediaSize : bigint
mediaSizeUncompressed : bigint
isApproved : boolean
user_id : bigint
performance_id : bigint
User SourceComment
SourceComment
id : bigint
description : text
createdAt : datetime
postedBy : string
postedByEmail : string
source_id : bigint
user_id : bigint
Source
ALL: GET
USER: POST
SOURCE: PATCH, DELETE
SourceComment
ALL: GET
USER: POST
SOURCE: DELETE
SourceChecksum
id : bigint
name : string
description : text
source_id : bigint
SourceChecksum
GUEST: GET
SOURCE: POST, PATCH, DELETE
SourceLink
ALL: GET
USER: POST
SOURCE: DELETE
SourceLink
id : bigint
name : string
url : text
source_id : bigint


api-skeletons/zf-oauth2-doctrine [\ZF\OAuth2\Doctrine\Entity]
Encryption [\ZF\OAuth2\Doctrine\Entity]
PublicKey
id : bigint
publicKey : text
privateKey : text
encryptionAlgorithm : string
client_id : bigint
Unique clientId
A unique index for clientId is required and added dynamically through the metadata subscriber. It is not in the schema to allow for custom naming strategies.
Client
id : bigint
clientId : string
secret : string
redirectUri : text
grantType : array
Jwt
id : bigint
subject : string
publicKey : text
client_id : bigint
Jti
id : bigint
subject : string
audience : string
expires : datetime
jti : text
client_id : bigint
Audience
A Jti is a one-time token. Per page 6 section 7 the OAuth2 server may maintain a list of Jwt which are used. Per https://github.com/bshaffer/oauth2-server-php/pull/268#issuecomment-71499663 the use marker may be tracked in audience.
Disconnected User
The \ZF\OAuth2\Entity\User does not exist and should be replaced with a custom user entity with a client relation to dynamically associate them. This is dynamically mapped and defined in the config.
Token [\ZF\OAuth2\Doctrine\Entity]
AuthorizationCode
id : bigint
authorizationCode : string
redirectUri : text
expires : datetime
idToken : text
client_id : bigint
AccessToken Counter
AccessToken
id : bigint
accessToken : text
expires : datetime
client_id : bigint
RefreshToken
id : bigint
refreshToken : string
expires : datetime
client_id : bigint
Scope Fixture
The Scope entity is intended for use as a fixture. You must manually assign an id to each scope.
Unique Scope name
A unique index for name is required and added dynamically through the metadata subscriber. It is not in the schema to allow for custom naming strategies.
Scope [\ZF\OAuth2\Doctrine\Entity]
ClientToScope
client_id : bigint
scope_id : bigint
Scope <--> Client
AuthorizationCodeToScope
authorization_code_id : bigint
scope_id : bigint
Scope <--> AuthorizationCode
AccessTokenToScope
access_token_id : bigint
scope_id : bigint
Scope <--> AccessToken
Scope
id : bigint
scope : string
isDefault : boolean
RefreshTokenToScope
refresh_token_id : bigint
scope_id : bigint
Scope <--> RefreshToken
Legend [\ZF\OAuth2\Doctrine\Entity]
Fixture
Comment
Author
Tom H Anderson
API Skeletons
<contact@apiskeletons.com>


Request Limiter [\RequestLimiter\Entity]
AccessToken Counter
Counter
id : integer
counter : bigint
epoch : datetime
updatedAt : datetime
access_token_id : bigint
Scope
RateLimit_1s
TimeLimit_86400s_2500
RateLimit_250ms
API Limiting
In order to limit the total number of daily requests and the rate at which requests are made, this module watches Scope[s] for Client for rules.
Each Counter row tracks a single AccessToken. If an request is made after the epoch expires, such as x / day, then the epoch is reset.
If a request is made too quickly based on updatedAt the request is put to sleep. (may not be the best solution)