Overview: This is a case study of Spotify, a commercial music-streaming service with an emphasis on being highly customizable, both for its users and developers. In order to facilitate modification Spotify has released a web API that allows developers access to a wide variety of information on its offerings, ranging from available markets to duration in milliseconds. This case study will focus on how this API facilitates access and interaction with the resources within Spotify from the developer's perspective. Spotify as an organizing system is highly complex. As a result, to better assist developers, the API does not assume an origin and instead takes a faceted organizing approaching, with robust resource description, to allow developers greater freedom in creating their own organizing systems. In addition to the case study, I have created an Entity Relational Diagram, which helps to illustrate the faceted nature of the API.

What is being organized?: This is ostensibly an easy question to answer, as Spotify is a music-streaming service. However, Spotify is organizing more than just music. Spotify incorporates a number of social elements into its system, such as shared playlists, user profile and even followers. For the purposes of this case study, I will focus exclusively on resources related to music. The main three resources that relate to music are: Artist, Album and Track. The availability of these resources is dependent on Spotify's agreements with different record labels. The agreements have a high level of granularity, even down to allowing certain tracks to be available in certain countries while not in others. In addition to the music itself, the Spotify API also organizes the representations of these resources (in JSON), which can be considered another type of resource in this system.

Why is it being organized?: The reason Spotify has created this API and exposed its data for developers is two-fold. Ultimately, the resources are being organized in order to help the user discover and access music within the system, but the resources are also being organized for developers who wish to aid in discovery and access. Instead of being prescriptive and using a hierarchical organizing structure, Spotify uses robust resource descriptions to create a faceted organizing system. This allows developers to create additional organizing systems that operate within Spotify. By allowing developers to create new organizing systems that work with Spotify, users are able to interact with the resources in new and novel ways that could not have been foreseen by Spotify. This gets to the heart of the second reason Spotify has organized its data and exposed it via the API:the more effective Spotify is at exposing its collection, the more it is able to differentiate itself from its competition. There are a number of other music providers and so simply providing access to music is insufficient to be competitive in the music-streaming market.

How much is it being organized?: The information from the Spotify web API is highly organized, allowing for developers to access information about resources in a great deal of granularity. The web API itself has thirty different types of RESTful methods available. Each of these methods represents a different, higher level, resource representations such as Album or Artist. Of these RESTful methods, the GET methods each return JSON documents that have additional resource descriptions that cover all major types of facets except for spectrum facets. This level of granularity is necessary in order to support developers because it is unclear exactly what organizing system a developer will create and subsequently it is impossible to foresee

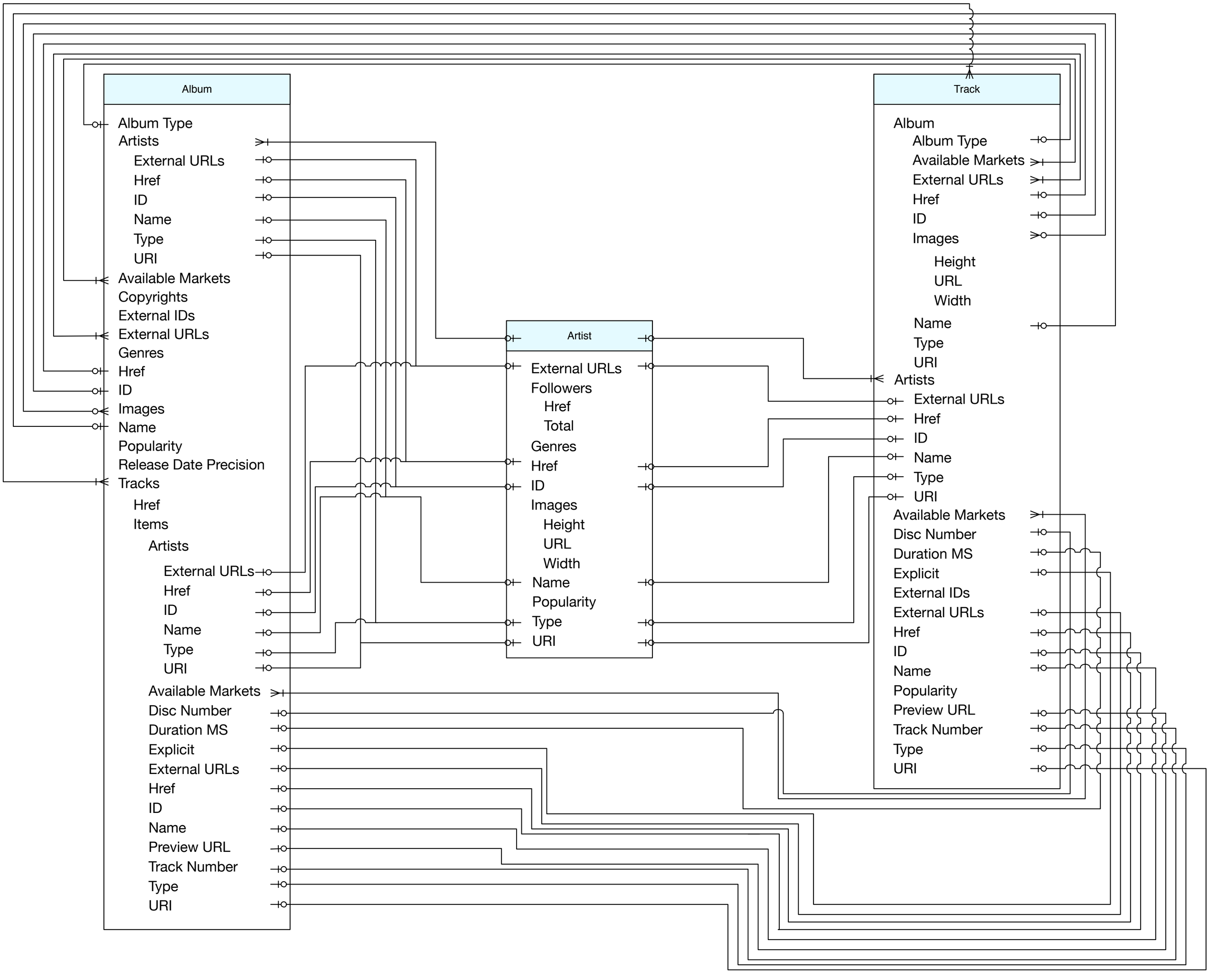
what information will be necessary. Additionally, resource descriptions are often repeated within the API depending on the perspective. An example of this is the song “The Robots” on Kraftwerk’s album “The Man Machine”. Because the API is designed to be faceted, it does not assume an origin and so will repeat information such as artist and duration for “The Robots” when making a GET requests on the track itself, or on the album. This repetition may seem inefficient, but it allows for developers to be more flexible as they create their own organizing system. By leveraging a faceted classification system, Spotify allows developers to make better use of its heterogeneous collection to create new interactions for the user.

When is it being organized?: The resources are being organized at two different points. The first is that the API is exposing information that Spotify has already internally added to the resource. The resource descriptions are typically created when the resource itself is added to the Spotify system. The second point at which the resources are being organized is when the GET requests being made by the developer. When the GET request is made, the Spotify API is selecting only the resource descriptions related to the specific GET request being made and returning this in a JSON. By organizing the resources in such a fashion, Spotify offers an extremely flexible way for developers to interact with the system.

How or by whom is it being organized?: The resources in the Spotify API are organized by two separate entities: by content owners (the record companies) and by Spotify itself. The content owners provide resource descriptions such as artist, album, and even available markets. These are the atomic resource descriptions that Spotify applies to the resources. In addition to the resource descriptions being provided by the content owners, there is also information being computationally generated by Spotify, such as the number of times played, and even the popularity of a resource. From this, Spotify then reorganizes the resource descriptions and applies them to the various representations for the API. Subsequently what is returned via JSON when a developer makes a GET request is a combination of human and computationally generated resource descriptions. In this way Spotify is able to mitigate the problems of using an enumerated hierarchical organizing structure.

Other considerations: Spotify also organizes quite a bit of social information about its users. While this information falls outside the scope of this case study, it is important to note that this is all accessible via the Spotify API. Developers could potentially utilize these additional facets in order to help facilitate discovery. Through machine learning and predictive programming practices, developers could identify similar users and expose users to music they are unfamiliar with, but would most likely enjoy.

Spotify API


Album
Album Type
Artists
External URLs
Href
ID
Name
Type
URI
Available Markets
Copyrights
External IDs
External URLs
Genres
Href
ID
Images
Name
Popularity
Release Date Precision
Tracks
Href
Items
Artists
External URLs
Href
ID
Name
Type
URI
Available Markets
Disc Number
Duration MS
Explicit
External URLs
Href
ID
Name
Preview URL
Track Number
Type
URI
Artist
External URLs
Followers
Href
Total
Genres
Href
ID
Images
Height
URL
Width
Name
Popularity
Type
URI
Track
Album
Album Type
Available Markets
External URLs
Href
ID
Images
Height
URL
Width
Name
Type
URI
Artists
External URLs
Href
ID
Name
Type
URI
Available Markets
Disc Number
Duration MS
Explicit
External IDs
External URLs
Href
ID
Name
Popularity
Preview URL
Track Number
Type
URI