amazon alexa

Complex Search and Play

Alexa Music Skills Kit
Developer Documentation

Version 2.0

Document version control

#	Documentation updates	Version number	Date
1	Intial document creation	1	5/13/2024
2	Add section for Matched Criteria	1.1	5/14/2024
3	Add capability framework	1.2	5/15/2024
4	Add confidence bins for ResolvedSelectionCriteria	1.3	6/10/2024
5	Clarify DEFAULT vs EXACT modifier. Remove EXACT_OR_SIMILAR	1.4	6/11/2024
6	Updated to finalized specification	2	7/30/2024

1. Overview:

Alexa Audio is expanding its Music Skills Kit (MSK) Search interface with Media Service Provider (MSP) skills to include more flexible query capabilities. These enhanced Search queries will enable customers to play and discover Audio content on Alexa using more expressive language.

2. Benefit:

With introduction of LLMs and new customer expectations from an AI assistant, Alexa anticipates customers will use more expressive and conversational language to discover (“suggest”, “recommend”, “find”) and listen (“play”) to music, radio and podcasts. Some examples that customers may expect their AI assistant to fulfill include:

1. **Multi-Entity requests -** Customers asking to “play songs by taylor swift, lady gaga and ed sheeran”
2. **Similar To requests -** Customers asking to “recommend a podcast similar to the one from wall street journal”
3. **Negation requests -** Customers asking to “play pop music, but nothing by taylor swift”
4. **Requests without clear catalog matches -** Customers asking “play rock songs with iconic guitar solos” where there is no existing mechanism to represent “iconic guitar solos” in existing MSK search queries.
5. **Multi-Turn Requests -** Customers engaging in open-ended conversations with Alexa to find content. For example, a customer is speaking to Alexa to plan a birthday party, and at some point say “can you recommend some music for the party?”

The updates to MSK mentioned in this document will enable the MSP skill to support these advanced use-cases. In addition to these MSK updates, Alexa will also provide the following features with no additional effort

needed from the MSP skill:

1. **Context-carryover** across multiple turns. Alexa will pull context from prior turns that is relevant to the MSP query. For example, if the user asks “who won the grammy for best album”, Alexa will answer with “Harry Styles won album of the year for his album Harry’s House”. If the user asks, “how old is he?”, Alexa will answer with “Harry styles is 30 years old”. If the user then follows up with “play songs from that album”, Alexa will query the MSP for “play songs from the album Harry’s House by Harry Styles”.
2. **Query Reformulation** across multiple turns involving the MSP. Alexa can compose a single query with information from multiple turns. For example, if the user asks “play pop music”, and then says “play something without taylor swift”, Alexa will query the MSP for “play pop music without taylor swift”

3. Implementation:

3.1. SUMMARY OF CHANGES

Alexa is updating the following SPIs under the `Alexa.Media.Search` namespace.

1. GetDisplayableContent
2. GetPlayableContent

A new version `3.0` will be introduced for these SPIs, and all changes discussed in this document will be under the Version `3.0` implementation.

For these SPIs, we are proposing several changes to extend the expressiveness of our search solution with MSPs, simplify the SPI structure, and reduce verbosity. Here is a summary of the changes being proposed:

1. Alongside the existing `ResolvedSelectionCriteriaAttributes` provided to the MSP skill to search for content, we will optionally provide a new `Query` structure to describe complex relationships between the Attributes. The mental model here is to think of Attributes as search terms captured from the user request, while the Query represents meaningful relationships such as combinations of attributes and negation of attributes.
2. To support the aforementioned `Query` structure, we will introduce a new set of classes called `Predicates`. A Query is composed of one or more Predicates.
3. For `Attributes` that convey resolved entities from an MSP skill catalog, we will extend the structure to allow multiple entity resolutions for a single named entity. i.e. if the user mentions "play songs by Taylor", the V3.0 SPI will provide a single Attribute containing resolutions for both "TaylorSwiftEntity" and "TaylorJamesEntity". These resolutions will be ranked in order of relevance, as determined by our Entity Resolution services.
4. The `ResolvedSelectionCriteria` structure will be made polymorphic to support search mechanisms that are different from our current slotted "Attribute" based search. We will introduce a new `NaturalLanguageSelectionCriteria` class to represent a free-form text search query which MSPs can use with their own LLMs.

Although the new SPI version has a simplified structure and less verbosity, it can convey much more complex search requests to our MSPs, and not all MSPs may be able to support all the `Attributes` and `Predicates` that we define. We will extend the schema of the skill manifest to allow MSP skills to declare fine-grained capabilities for `Attributes` and `Predicates.`

3.2. IDENTIFIERS FOR RESOLVEDSELECTIONCRITERIAATTRIBUTES

We are adding identifiers to every `ResolvedSelectionCriteriaAttribute` object. This allows Attributes to be referenced by other building blocks in the SPI request.

The base structure of `ResolvedSelectionCriteriaAttribute` will now have the following fields:

	Field	Description	New in V3.0?	Required?	Type
1	type	Defines the type of the attribute	No	Yes	String representing the EntityType
2	id	An identifier for this attribute.	Yes	Yes	String

3.3. MULTIPLE ENTITY RESOLUTIONS FOR CATALOG ATTRIBUTES

`ResolvedSelectionCriteriaAttribute` objects with entity types ALBUM, ARTIST, GENRE, PLAYLIST, STATION, TRACK, PROGRAM_SERIES, PROGRAM, RADIO, BOOK, or AUTHOR are used to model resolved entities from an MSPs catalog. The current structure allows us to specify a single entity resolution in each object. We will modify the specification to have multiple resolved entities in a single object. If a user mentions an entity that cannot be unambiguously resolved to a single catalog identifier, Alexa may provide a ranked list of such resolutions to the skill.

The primary motivation here is efficiency and reduced verbosity. When ambiguous entity resolutions exist today, the current system creates multiple `ResolvedSelectionCriteria` objects, with each Criteria object representing one possible permutation of Entities resolved for each Attribute. This grows exponentially with the number of Attributes and Entity resolutions, creating an unmanageable amount of combinations.

Note the difference between:

1. Having multiple resolutions for a single identified Entity If the user says "play songs by Taylor ", A single `ResolvedSelectionCriteriaAttribute` ARTIST object will contain all resolutions for "Taylor", including a resolution for "Taylor Swift" and another resolution for "Taylor James".
2. Having multiple identified entities: If the user says "play songs by Taylor and Ed Sheeran", A single `ResolvedSelectionCriteriaAttribute` ARTIST object will contain all resolutions for "Taylor" and another `ResolvedSelectionCriteriaAttribute` ARTIST object will contain all resolutions for "Ed Sheeran".

	Field	Description	New in V3.0?	Required?	Type
1	type	Defines the type of the attribute	No	Yes	String representing the EntityType
2	id	An identifier for this attribute.	Yes	Yes	String
3	resolvedEntities	Conveys resolved entities for a single slot value, sorted by relevance defined by the ER system.	Yes	Yes	Array
4	resolvedEntities[i]	Conveys a single resolution for a single slot value	Yes	Yes	Object
5	resolvedEntities[i].entityId	Skill-provided identifier for the piece of content this attribute represents. This identifier gets ingested as part of one of the skill's catalogs.	Yes	Yes	String

3.3. PREDICATE CLASSES

For complex utterances, we need mechanisms to compose multiple Attributes into a complex query. For example, a request to "play songs similar to taylor swift but not by ed sheeran" can be thought of as a query operation of

(MediaType=Songs) INTERSECTION SIMILAR(Artist=TaylorSwift) INTERSECTION NOT(Artist=EdSheeran).

To achieve this, we will define a new set of classes called `Predicates`. Predicates can be chained together to form an N-ary tree of Predicates, with the leaf nodes of the tree referencing Attributes to be used to search for Content. We define 3 types of Predicates:

1. NaryQueryPredicate (Internal Tree Node): An internal node that applies a condition to two or more sub-predicates. Acts as a reduction operator to combine multiple predicates together to a single result.
2. UnaryQueryPredicate (Internal Tree Node): An internal node that applies a condition a single sub-predicate. Acts as a decorator that modifies the results of the sub-predicate.
3. AttributeQueryPredicate (Leaf Node): A reference to a ResolvedSelectionCriteriaAttribute class (for example, an artist, album, podcast series, sort type, or position of an item in a list)

Taking the previous example of "play songs similar to taylor swift but not by ed sheeran", this can be represented in a Predicate Tree as follows:

3.3.1 Base Structure - SelectionQueryPredicate

A base class for all Query Predicates. Specifies the type of the Predicate to deserialize as appropriate.

	Field	Description	New in V3.0?	Required?	Type
1	type	The concrete type of this Predicate. Allowed values are: NARY UNARY ATTRIBUTE	Yes	Yes	String. Allowed Values: - `NARY` - `UNARY` - `ATTRIBUTE`

3.3.2 NaryQueryPredicate

A predicate that applies a condition to a list of predicates. Acts similar to a reduction operator and defines how the MSP skill should combine multiple predicates to a single result as part of their content search.

	Field	Description	New in V3.0?	Required?	Type
1	type	The type of this predicate. Value will be **NARY**	Yes	Yes	String
2	condition	The condition to apply when combining the list of predicates to a single result. Supports the following: 1. UNION - The content returned must be based on items that satisfy one or more of the sub-predicates. Each returned item can satisfy any sub-predicate(s). The MSP skill should attempt to ensure that each sub-predicate is satisfied by atleast one returned content item. For example, if two sub-predicates P1 and P2 are supplied, items can be drawn from a combined pool of items that either satisfy both P1 and P2, satisfy only P1, or satisfy only P2. 2. INTERSECTION - The content returned must be based on items that satisfy all sub-predicates. Each returned item must satisfy all sub-predicates. For example, if two sub-predicates P1 and P2 are supplied, returned items must be drawn from a pool of items that satisfy both P1 and P2.	Yes	Yes	String. Allowed Values: - `UNION` - `INTERSECTION`
3	predicates	List of predicates to combine	Yes	Yes	Array of `SelectionQueryPredicate`

3.3.3 UnaryQueryPredicate

A predicate that applies a condition to another predicate. Acts as a decorator that modifies the results of the sub-predicate.

	Field	Description	New in V3.0?	Required?	Type
1	type	The type of this predicate. Value will be **UNARY**	Yes	Yes	String
2	condition	The condition to apply on the sub-predicate. Supports the following: 1. SIMILAR - The content returned must be based on items that are similar to the items that satisfy the sub-predicate. 2. NOT - The content returned must not include items that satisfy the sub-predicate.	Yes	Yes	String. Allowed Values: - `SIMILAR` - `NOT`
3	predicate	The predicate to decorate.	Yes	Yes	`SelectionQueryPredicate`

3.3.4 AttributeQueryPredicate

A predicate to reference an Attribute by its identifier.

	Field	Description	New in V3.0?	Required?	Type
1	type	The type of this predicate. Value will be **ATTRIBUTE**	Yes	Yes	String
2	attributeId	The identifier of the Attribute to use.	Yes	Yes	String

3.4. POLYMORPHIC SELECTION CRITERIA

Existing versions of our SPI only allow for searching content through "slots" or "attributes". These are rigidly defined entities that match attributes present in MSP skill catalogs. The SPI needs to support additional strategies to search for content. To support this, we will modify our `ResolvedSelectionCriteria` object to be polymorphic, with each concrete implementation describing a different search strategy.

	Field	Description	New in V3.0?	Required?	Type
1	type	The concrete type of this Selection Criteria. Allowed values are: ATTRIBUTES NL_QUERY	Yes	Yes	String. Allowed Values: - `ATTRIBUTES` - `NL_QUERY`
2	id	A unique identifier for this Selection Criteria object	Yes	Yes	String

3.4.1 MultiAttributeSelectionCriteria

MultiAttributeSelectionCriteria is the V3.0 equivalent to the slotted representations we currently provide in our SPIs. In addition to the fields present in previous versions, we also add an optional `Query` composed of `Predicate` classes, and a score that informs the MSP how well this Selection Criteria captures the customer intent. The MSP may choose to use this score when deciding whether to utilize the MultiAttributeSelectionCriteria or pick a different SelectionCriteria object.

	Field	Description	New in V3.0?	Required?	Type
1	type	The concrete type of this Selection Criteria. Will be set to **ATTRIBUTES**	Yes	Yes	String
2	id	A unique identifier for this Selection Criteria object	Yes	Yes	String
3	attributes	A list of ResolvedSelectionCriteriaAttribute objects to be used to search for Content. This list only enumerates various search terms captured from the user request, but does not describe the relationship between these search terms. The relationship between Attribute objects is defined by the `query` field. If no `query` is specified, the default behavior is to apply an INTERSECTION operation to all Attribute objects. Specifically, this means that the content returned must be based on items that meet all provided Attributes.	No	Yes	Array of `ResolvedSelectionCriteriaAttribute`
4	query	An expression that defines how the ResolvedSelectionCriteriaAttribute objects from the `attributes` field should be combined to satisfy the user's request. The query may consist of a single predicate, or multiple predicate organized in an N-ary Tree structure. The leaf nodes of this tree will always be an AttributeQueryPredicate, referencing one of the Attributes defined in the `attributes` field. If no `query` is specified, the default behavior is to apply an INTERSECTION operation to all Attribute objects. Specifically, this means that the content returned must be based on items that meet all provided Attributes.	Yes	No	`SelectionQueryPredicate`
5	completeness	Describes how completely this selection criteria captures customer intent. If a `query` is provided, the completeness score encapsulates the information contained in both the `attributes` and `query` field when taken together. The score is not based on the information contained in `attributes` in isolation. It is assumed that the MSP skill will utilize both `attributes` and `query` to fulfill the customer intent. If no `query` is provided, the completeness score is based on the information in `attributes`, assuming the default behavior of applying an INTERSECTION operation across all Attributes.	Yes	No	Object
6	completeness.score	A raw score ranging from 0 to 1, with 1 representing complete capture of the customer intent and 0 representing complete loss of customer intent.	Yes	No	Float
7	completeness.bin	A bin that classifies how well the customer intent has been captured. HIGH implies that the customer intent has been modeled completely. MEDIUM implies that significant portions of the customer intent have been captured, but some nuance may be lost. LOW implies that there is significant information loss from the original customer intent.	Yes	Yes	String. Allowed Values: - `HIGH` - `MEDIUM` - `LOW`

3.4.2 NaturalLanguageSelectionCriteria

A `NaturalLanguageSelectionCriteria` object represents a free-form text query sent to the MSP skill with no defined entities or relationships. The free-form text query is not the same as the raw customer utterance, but is generated by Alexa based off the raw customer utterance. It will exclude information that is not relevant to the MSP, such as any personally identifiable information. The primary advantage of a natural language query is in preserving information from the customer utterance without reliance on any rigid modeling or attribute schemas.

	Field	Description	New in V3.0?	Required?	Type
1	type	The concrete type of this Selection Criteria. Will be set to **NL_QUERY**	Yes	Yes	String
2	id	A unique identifier for this Selection Criteria object	Yes	Yes	String
3	query	The free-form text query to be used by the MSP to search their catalog.	Yes	Yes	String

3.5. MATCHEDCRITERIA

A `MatchedCriteria` object is included in the Response from the MSP and identifies the selection criteria used by the MSP skill to search for Content. This is important for maintaining transparency with the customer because its possible that an MSP may choose to use a specific MultiAttributeSelectionCriteria which may not completely model the customer request.

	Field	Description	New in V3.0?	Required?	Type
1	criteriaId	The identifier of the ResolvedSelectionCriteria used by the MSP skill to resolve Content for this request.	Yes	Yes	String

3.6. SPI REQUEST AND RESPONSE PAYLOADS

In Version 3.0, we modify the request/response payloads for GetPlayableContent and GetDisplayalbeContent SPIs. The modified SPI payloads will use the building blocks outlined above. The sections below outline the structure of each of these payload objects, noting where things have changed.

3.6.1 GetPlayableContent Request Payload

	Field	Description	New in V3.0?	Required?	Type
1	~~selectionCriteria~~	Removed in V3.0	Removed in V3.0	Removed in V3.0	Removed in V3.0
2	rankedSelectionCriteria	A ranked list of ResolvedSelectionCriteria objects, where each ResolvedSelectionCriteria represents one possible interpretation of search criteria that should be resolved to one or more Content objects requested by the user. Note that Alexa's confidence in each object is implied by its position in the list. In other words, the list is sorted in descending order by confidence.	No	Yes	Array of `ResolvedSelectionCriteria`
3	filters	Unchanged from previous versions	No	Yes	Unchanged from previous versions
4	requestContext	Unchanged from previous versions	No	Yes	Unchanged from previous versions

3.6.2 GetPlayableContent Response Payload

	Field	Description	New in V3.0?	Required?	Type
1	content	Unchanged from previous versions	No	Yes	Unchanged from previous versions
2	matchedCriteria	The identifier of the `ResolvedSelectionCriteria` object used by the MSP to resolve Content for this request	Yes	No	`MatchedCriteria` object

3.6.3 GetDisplayableContent Request Payload

	Field	Description	New in V3.0?	Required?	Type
1	~~alexaResolvedSelectionCriteria~~	Removed in V3.0	Removed in V3.0	Removed in V3.0	Removed in V3.0
2	rankedSelectionCriteria	A ranked list of ResolvedSelectionCriteria objects, where each ResolvedSelectionCriteria represents one possible interpretation of search criteria that should be resolved to one or more Content objects requested by the user. Note that Alexa's confidence in each object is implied by its position in the list. In other words, the list is sorted in descending order by confidence.	Yes	Yes	Array of `ResolvedSelectionCriteria`
3	maxResultLimit	Unchanged from previous versions	No	Yes	Unchanged from previous versions
4	playQueuePreviewCriteria	Unchanged from previous versions	No	No	Unchanged from previous versions
5	endpoints	Unchanged from previous versions	No	No	Unchanged from previous versions
6	filters	Unchanged from previous versions	No	Yes	Unchanged from previous versions
7	requestContext	Unchanged from previous versions	No	Yes	Unchanged from previous versions

3.6.4 GetDisplayableContent Response Payload

	Field	Description	New in V3.0?	Required?	Type
1	contentGroups	Unchanged from previous versions	No	Yes	Unchanged from previous versions
2	matchedCriteria	The identifier of the `ResolvedSelectionCriteria` object used by the MSP to resolve Content for this request	Yes	No	`MatchedCriteria` object

3.7. SKILL MANIFEST

Given the scope of the changes described above, Alexa expects that different MSP skills may develop capabilities for handling natural language queries and complex predicates at different times. To solve this problem, Alexa is introducing a mechanism for MSP skills to declare fine-grained capabilities for the SPIs under `Alexa.Media.Search` namespace. MSP skills are expected to update their skill manifest to list the Criteria, Attributes, and Predicates they support. Because some of the new objects we introduce can significantly change the semantics of the search query, Alexa needs to ensure it does not send objects that the MSP skill cannot interpret. As an example, if an MSP skill does not support a “NOT” modifier, Alexa should not expect the MSP skill to support a search query for “Not Taylor Swift”.

The existing skill manifest follows this structure

```
{
```

```
    "manifest":
    {
        "apis":
        {
            "music":
            {
                "endpoint": "...",
                "interfaces":
                [
                    {
                        "namespace": "Alexa.Media.Search",
                        "version": "2.0",
                        "requests":
                        [
                            {
                                "name": "GetPlayableContent"
                            }
                        ]
                    }
                ]
            }
        }
    }
}
```

The new skill manifest will include the ability to declare configuration for each onboarded SPI, like below:

```
{
    "manifest":
    {
        "apis":
        {
            "music":
            {
                "endpoint": "...",
                "interfaces":
                [
                    {
                        "namespace": "Alexa.Media.Search",
                        "requests":
                        [
                            {
                                "name": "GetPlayableContent",
                                "version": "3.0",
                                "configuration":
                                {
                                    "version": "1.0",
                                    "supportedCriteria":
                                    [
                                        {
                                            "type": "NL_QUERY"
                                        },
                                        {
                                            "type": "ATTRIBUTES",
```

```
                                        "supportedAttributes":
                                        [
                                            {
                                                "type": "ARTIST"
                                            },
                                            {
                                                "type": "GENRE"
                                            },
                                            {
                                                "type": "MEDIA_TYPE"
                                            }
                                        ],
                                        "unaryQueryConditions":
                                        [
                                            "SIMILAR",
                                            "NOT"
                                        ],
                                        "naryQueryConditions":
                                        [
                                            "INTERSECTION",
                                            "UNION"
                                        ]
                                    }
                                ]
                            }
                        }
                    ]
                }
            ]
        }
    }
}
```

4. Examples

4.1. "ALEXA, PLAY SONGS SIMILAR TO TAYLOR SWIFT, BUT NO ED SHEERAN PLEASE"

The object below represents a sample GPC V3.0 request payload using both the new NaturalLanguageSelectionCriteria and Unary/Nary Predicate classes.

```
{
    "header":
    {
        "messageId": "...",
        "namespace": "Alexa.Media.Search",
        "name": "GetDisplayableContent",
        "payloadVersion": "3.0"
    },
    "payload":
    {
        "requestContext": "...",
        "filters": "...",
        "rankedSelectionCriteria":
```

```
[
    {
        "id": "Criteria1",
        "type": "NL_QUERY",
        "query": "songs similar to taylor swift but no ed sheeran"
    },
    {
        "id": "Criteria2",
        "type": "ATTRIBUTES",
        "attributes":
        [
            {
                "id": "MediaTypeAttr1",
                "type": "MEDIA_TYPE",
                "value": "SONGS"
            },
            {
                "id": "ArtistAttr1",
                "type": "ARTIST",
                "resolvedEntities":
                [
                    {
                        "entityId": "taylor entity 1"
                    }
                ]
            },
            {
                "id": "ArtistAttr2",
                "type": "ARTIST",
                "resolvedEntities":
                [
                    {
                        "entityId": "ed sheeran 1"
                    }
                ]
            }
        ],
        "query":
        {
            "type": "NARY",
            "condition": "INTERSECTION",
            "predicates":
            [
                {
                    "type": "ATTRIBUTE",
                    "attributeId": "MediaTypeAttr1"
                },
                {
                    "type": "UNARY",
                    "condition": "SIMILAR",
                    "predicate":
                    {
                        "type": "ATTRIBUTE",
                        "attributeId": "ArtistAttr1"
                    }
```

```
                        },
                        {
                            "type": "UNARY",
                            "condition": "NOT",
                            "predicate":
                            {
                                "type": "ATTRIBUTE",
                                "attributeId": "ArtistAttr2"
                            }
                        }
                    ]
                },
                "completeness":
                {
                    "score": 1,
                    "bin": "HIGH"
                }
            }
        ]
    }
}
```

4.2. "ALEXA, PLAY TAYLOR SWIFT SONGS WITH ICONIC GUITAR SOLOS"

This example demonstrates an MSP that supports a NaturalLanguageSelectionCriteria, but does not declare support for any predicates. The query field in MultiAttributeSelectionCriteria is omitted and a default INTERSECTION operation is expected to apply to all Attributes. The completeness score of the MultiAttributeSelectionCriteria is reduced because there is no way to represent "iconic guitar solos" in the MSP skill catalog.

```
{
    "header":
    {
        "messageId": "...",
        "namespace": "Alexa.Media.Search",
        "name": "GetDisplayableContent",
        "payloadVersion": "3.0"
    },
    "payload":
    {
        "requestContext": "...",
        "filters": "...",
        "rankedSelectionCriteria":
        [
            {
                "id": "Criteria1",
                "type": "NL_QUERY",
                "query": "taylor swift songs with iconic guitar solos"
            },
            {
                "id": "Criteria2",
                "type": "ATTRIBUTES",
                "attributes":
                [
```

```
                    {
                        "id": "MediaTypeAttr1",
                        "type": "MEDIA_TYPE",
                        "value": "SONGS"
                    },
                    {
                        "id": "ArtistAttr1",
                        "type": "ARTIST",
                        "resolvedEntities":
                        [
                            {
                                "entityId": "taylor entity 1"
                            }
                        ]
                    }
                ],
                "completeness":
                {
                    "score": 0.4,
                    "bin": "MEDIUM"
                }
            }
        ]
    }
}
```

Presumably the MSP skill used the NaturalLanguageSelectionCriteria to fulfill the request, and they should indicate so in their response

```
{
    "header":
    {
        "messageId": "...",
        "namespace": "Alexa.Media.Search",
        "name": "GetDisplayableContent.Response",
        "payloadVersion": "3.0"
    },
    "payload":
    {
        "content": "...",
        "matchedCriteria":
        {
            "criteriaId": "Criteria1"
        }
    }
}
```

4.3. PLAY SONGS BY ARTISTS SIMILAR TO TAYLOR SWIFT AND ED SHEERAN“

The example below demonstrates how Unary and Nary Predicates can be chained together

```
{
    "header":
    {
```

```
        "messageId": "...",
        "namespace": "Alexa.Media.Search",
        "name": "GetDisplayableContent",
        "payloadVersion": "3.0"
    },
    "payload":
    {
        "requestContext": "...",
        "filters": "...",
        "rankedSelectionCriteria":
        [
            {
                "id": "Criteria1",
                "type": "NL_QUERY",
                "query": "songs by artists similar to taylor swift and ed sheeran"
            },
            {
                "id": "Criteria2",
                "type": "ATTRIBUTES",
                "attributes":
                [
                    {
                        "id": "MediaTypeAttr1",
                        "type": "MEDIA_TYPE",
                        "value": "SONGS"
                    },
                    {
                        "id": "ArtistAttr1",
                        "type": "ARTIST",
                        "resolvedEntities":
                        [
                            {
                                "entityId": "taylor entity 1"
                            }
                        ]
                    },
                    {
                        "id": "ArtistAttr2",
                        "type": "ARTIST",
                        "resolvedEntities":
                        [
                            {
                                "entityId": "ed sheeran 1"
                            }
                        ]
                    }
                ],
                "query":
                {
                    "type": "NARY",
                    "condition": "INTERSECTION",
                    "predicates":
                    [
                        {
                            "type": "ATTRIBUTE",
```

```
                            "attributeId": "MediaTypeAttr1"
                        },
                        {
                            "type": "UNARY",
                            "condition": "SIMILAR",
                            "predicate":
                            {
                                "type": "NARY",
                                "condition": "UNION",
                                "predicates":
                                [
                                    {
                                        "type": "ATTRIBUTE",
                                        "attributeId": "ArtistAttr1"
                                    },
                                    {
                                        "type": "ATTRIBUTE",
                                        "attributeId": "ArtistAttr2"
                                    }
                                ]
                            }
                        }
                    ]
                },
                "completeness":
                {
                    "score": 1,
                    "bin": "HIGH"
                }
            }
        ]
    }
}
```

Appendix

Raw Slot Values

In version 3.0, the existing `RawSelectionCriteria` and `RawSelectionCriteriaAttribute` objects are deprecated. Raw slot value are now inlined within the `ResolvedSelectionCriteriaAttribute` object. Inlining is a simpler solution as we do not need to maintain two separate structures of RawSelectionCriteria and ResolvedSelectionCriteria which need to be kept in sync. Since we plan to increase the complexity of ResolvedSelectionCriteriaAttributes to hold multiple entity resolutions, the complexity of keeping the two structures in sync will increase as well.

An example `ResolvedSelectionCriteriaAttribute` with inlined raw slot value is shown below:

```
{
   "id": "ArtistAttr1",
   "type": "ARTIST",
   "rawValue": "Taylor", // Replacement for RawSelectionCriteria
   "resolvedEntities":
   [
```

```
        {
            "entityId": "taylor entity 1"
        }
    ]
}
```