Alexa Music Personal Playlists

Alexa Music Skills Kit Developer External Documentation

Version 1.0

AMAZON CONFIDENTIAL

Table of Contents

1 Overview

Alexa music supports personal playlist playback internationally, which allows end customers to use their Alexa voice assistant to search for and playback personal playlists and stations from their favorite music providers worldwide. As a music provider, this document outlines how to provide asynchronous catalog updates to Alexa so you can send user-generated playlist and station updates to Alexa if/when end customers make updates to their playlists and stations from your app or website. Please note that there is a limit of 2500 personalized catalog entities per end customer/user. If you require support for additional entity types or personal catalog size, please reach out to your Alexa Music business representative.

This feature currently supports playback of the following entity types:

- *Playlist*
- *Station*

We are including a preview of an unreleased Music Skill API in this document. Because this is a preview, there may be changes to the API specification before it is supported in production.

2 Customer experience

Customers may now use Alexa to search for, play, and control their personal playlists and stations from their favorite music providers. "Alexa, play my Road Trip playlist" will queue up the customer's self-made playlist of their most-loved road trip songs. On multi-modal devices, customers can also ask, "Alexa, show me my <provider> stations" to browse through their personal stations. Customers can then use Alexa to voice control their experience while listening, including asking her to pause, resume, play next, or play previous. If you support SetLoop and SetShuffle, customers can also shuffle and loop their personal playlists and stations to switch up their listening experience.

3 Terminology

Term	Description
previousSyncToken	field used in the update/initial sync as well as in the Self-Healing process: * usage in update -- Helps identify the order of the update. The value to this field will be the syncToken value of the previous update. This would help Alexa detect if all updates prior to this update are received and processed by Alexa. * usage in Self-Heal process -- is the latest SyncToken that Alexa has for this user. During the self-heal process, this would be used by the partner to detect if Alexa is out-of-sync for this user * previousSyncToken = null -- indicates that this is a full catalog update for the user and the provided catalog information will be considered as the full snapshot of the catalog for the user.
syncToken	generated only by the provider and sent with the update/initial sync. Used in the Self-Heal process as previousSyncToken
correlationToken	generated only by Alexa. Alexa initiated requests would use this field to be able to correlate requests to async updates. When updates are sent as a response to an Alexa initiated-request, the correlationToken (from the request) should be included in the update
Skill Events	A skill event is generated whenever a customer enables or disables the skill, links or unlinks a third-party account with the skill, grants permissions to a skill, or changes the permissions grant to the skill. More info here
Skill Event Lambda	Lambda registered to receive Skill Events, can be the same or different Lambda function as the Music Skill Lambda. More info here
Skill Lambda	Lambda that implements support for the Music Skill APIs, can be the same or different Lambda function as Skill Event Lambda
userId	Amazon generated user ID
accessToken	Provider generated accessToken

4 Configure Manifest

If you are an existing provider and developing support for Async Updates:

1. Contact your Alexa Music representative and provide them with the skill ID, the vendor ID of your Amazon Developer account, and the customer ID of the Alexa user that you want to use to test personal catalogs.
2. If you haven't yet, please familiarize yourself with the Alexa Skills Kit Command Line Interface (ASK CLI). You will need to use this to make updates to your existing skill, and these updates cannot be done via the Developer Console.
3. Using the ASK CLI's get-skill command, get the skill manifest JSON for your existing Music Skill.
4. Add the following JSON snippet to your skill manifest as a property under manifest. You can see an example of this below and here.

```
"events": {
    "endpoint": {
       "uri": "your-skill-events-lambda-uri"
    },
```

```
    "regions": {
      "NA": {
        "endpoint": {
          "uri": "your-skill-events-lambda-uri"
        }
      }
    },
    "subscriptions": [
      {
        "eventName": "SKILL_DISABLED"
      },
      {
        "eventName": "SKILL_ACCOUNT_LINKED"
      }
    ]
  }
```

5. Use the ASK CLI's update-skill command to update your Music skill in development stage with your updated skill manifest JSON file

5 Scenarios

5.1 Prerequisites

1. You must register your Skill to receive SkillAccountLinked event by following instructions here.
2. Get clientId, clientSecret from your Alexa representative (see Next Steps below).
3. Get the S3 bucket for the skill from your Alexa Music representative.
4. Onboard to Login with Amazon (LWA).

5.2 Steps

1. Skill Event Lambda receives SkillAccountLinked event which contains your generated accessToken for the user and Amazon userId.
2. You update the mapping from accessToken to Amazon userId and stores in your service persistence.
1. International: Depending on the Amazon user's account region, the apiEndpoint property in the SkillAccountLinked event will be different, and the provider (you) will need to store the reference and relationship in their persistence as well
2. If the user is from North America (NA), e.g. CA, US, MX, BR, etc., then the apiEndpoint = https://api.amazonalexa.com
3. If the user is from Europe (EU), e.g. GB, IE, FR, DE, AT, ES, IT, IN (India), then the apiEndpoint = https://api.eu.amazonalexa.com
4. If the user is from Far East (FE), e.g. JP, AU, NZ, then the apiEndpoint = https://api.fe.amazonalexa.com

3. Upload the personal catalog for the user to S3 bucket (setup as part of skill creation) under the intake-user-catalog folder. Be sure to grant read/write to alexa-music-search+8@amazon.com for the uploaded files, follow instructions here
4. Exchange clientId, secret, scope alexa::music:catalog:write with LWA's token endpoint to get consumerToken
1. International: You can (but is not required to) call a different LWA endpoint to get the consumer token based on the region where their user-related service is located, for latency efficiency purposes:
2. If the user-related service is located in the Americas, then the LWA token endpoint is https://api.amazon.com/auth/O2/token
3. If the user-related service is located in the Europe and India, then the LWA token endpoint is https://api.amazon.co.uk/auth/O2/token
4. If the user-related service is located in the East Asia or Oceania, then the LWA token endpoint is https://api.amazon.co.jp/auth/O2/token
5. Receive consumerToken from LWA
6. Call the API https://{apiEndpoint}/v1/music/users/{userid}/catalog/sync with S3 URL, userId, consumerToken, where {apiEndpoint} is from the SkillAccountLinked event for this user
7. Alexa authenticates using consumerToken, validates userId
8. Alexa processes the personal catalog asynchronously
9. For skill re-linking, the same scenario will occur, however, only the new userId shared in the latest AccountLinked event will be valid

5.3 JSON File

```
{
   "correlationToken": "TreGEchefasWAVususuwAsTeP8",
   "syncToken": "bRAgUphUchA4ReYab28r",
   "previousSyncToken": null,
   "stations": [...],
   "playlists": [...]
}
```

5.3.1 JSON File Fields

Field	Description	Type	Required
correlationToken	see Terminology above	String	No
syncToken	see Terminology above	String	Yes
previousSyncToken	see Terminology above	String	Yes
stations	array of Station entities	Array	No
playlists	array of Playlist entities	Array	No

5.3.2 JSON File Mandatory Fields

Note that the entity JSONs are the same as the public catalogs, but for personal catalogs, the only mandatory properties for each entity are:

- id

- names
- deleted
- lastUpdatedTime

5.4 Sync API Request

```
POST/v1/music/users/{id}/catalog/sync -H 'Authorization: bearer consumerToken' -H 'Content-Type: application/json'

{

  "contentURL": "S3_URL"

}
```

5.4.1 Sync API Request Parameters

Field	Description	Type	Required
HEADER: Authorization	bearer {consumerToken}	String	Yes
HEADER: Content-Type	application/json	String	Yes
POST: id	user id from Alexa	String	Yes
BODY: contentURL	URL for the S3 file where personal catalog was uploaded	String	Yes

5.4.2 Sync API Responses

- 202 - Request Accepted. Validation of authorizationToken, userId, schema succeeded. No validation of catalog content is done at this point
- 400 - Invalid Request. Schema validation failed or the payload size is larger than 200KB
- 403 - Forbidden. consumerToken invalid/expired, Expired userId.
- 429 - Throttled, try again with exponential backoff
- 500 - Internal Service Error, try again after a while

6 Customer Updates (Update API)

6.1 Prerequisites

1. You must map your user to the Alexa user using the SkillAccountLinked event
2. You must have the clientId, clientSecret for the skill & the LWA scope to be used from Alexa Music
3. Onboard to LWA

6.2 Steps

1. Exchange you clientId, clientSecret and scope alexa::music:catalog:write to get a consumerToken
    a. International: You can (not required) call a different LWA endpoint to get the consumer token based on the region where their user-related service is located, for latency efficiency purposes:
    b. If the user-related service is located in the Americas, then the LWA token endpoint is https://api.amazon.com/auth/O2/token
    c. If the user-related service is located in the Europe and India, then the LWA token endpoint is https://api.amazon.co.uk/auth/O2/token

    d. If the user-related service is located in the East Asia or Oceania, then the LWA token endpoint is https://api.amazon.co.jp/auth/O2/token
2. Publish your update by calling the API https://{apiEndpoint}/v1/music/users/{userId}/catalog/update and identifies user using userId in the update
3. Max payload size for the API is 200 KB
4. Include at most 10 entities per update
5. Alexa is only interested in updates to entity names, not changes to tracks in personal playlists
6. Alexa authenticates using consumerToken, validates userId
7. Alexa processes the update asynchronously
8. Error detection – Alexa will verify the input and respond back with the adequate error response. (Similar responses to 5.4.2)
9. If Alexa detects that the previousSyncToken in this update doesn't match the latest sync token for the user (prior to the update) then Alexa optimistically processes the update while waiting (with a 300 second timeout) for the missing update (to accommodate out-of-order delivery of updates)
10. If the missing update didn't arrive then, Alexa will trigger the Self Healing process (described below) with previousSyncToken=null to re-sync the entire personal catalog for the user

6.3 Update API Request (for V2)

```
POST /v1/music/users/{id}/catalog/update -H 'Authorization: bearer consumerToken' -H 'Content-Type: application/json'
{
  "syncToken": "bRAgUphUchA4ReYab28r",
  "previousSyncToken": "weXAkeWuwrEmuvaREx8tH",
  "stations": [
    {
      "entityId": "1233456789",
      "name": [
        {
          "language": "en",
          "name": "Popular Music"
        }
      ],
      "alternateName": [
        {
          "language": "en",
          "alias": [
            "Popular Hits"
          ]
        }
      ],
      "languageOfContent": [
        "en"
      ],
      "channelId": 123,
      "callSign": "WXPR",
      "frequency": "95.5",
      "stationType": "LIVE",
      "popularity": 98,
      "ownedAndOperated": "No",
      "explicitContent": false,
      "deleted": false,
      "startDate": "2017-01-01T00:00:00Z",
      "endDate": "2020-01-01T00:00:00Z",
      "lastUpdatedTime": "2017-12-29T00:00:00Z"
    }
```

```
  ],
  "playlists": [
    {
      "entityId": "1233455555",
      "name": [
        {
          "language": "en",
          "name": "Best Hits of the 90s"
        }
      ],
      "languageOfContent": [
        "en"
      ],
      "explicitContent": true,
      "deleted": false,
      "startDate": "2017-01-01T00:00:00Z",
      "lastUpdatedTime": "2017-12-29T00:00:00Z"
    }
  ]
}
```

6.4 Update API Request Parameters

Field	Description	Type	Required
HEADER: Authorization	bearer {consumerToken}	String	Yes
HEADER: Content-Type	application/json	String	Yes
POST: id	user id from Alexa	String	Yes
BODY: syncToken	newly generated sync token for this update	String	Yes
BODY: previousSyncToken	sync token for the previous update	String	Yes
BODY: stations	see S3 File above	Array	No
BODY: playlists	see S3 File above	Array	No

6.5 Update API Request (for v3)

```
POST /v1/music/users/{id}/catalog/update -H 'Authorization: bearer consumerToken' -H 'Content-Type: application/json'

{
  "syncToken": "bRAgUphUchA4ReYab28r",
  "previousSyncToken": "weXAkeWuwrEmuvaREx8tH",
  "stations": [
    {
      "id": "1233456789",
      "names": [
```

```
            {
                "language": "en",
                "value": "Popular Music"
            }
        ],
        "alternateNames": [
            {
                "language": "en",
                "values": [
                    "Popular Hits"
                ]
            }
        ],
        "locales": [
            {
                "country": "US",
                "language": "en"
            }
        ],

        "channelId": 123,
        "callSign": "WXPR",
        "frequency": "95.5",
        "type": "LIVE",
        "popularity":
            {
                 "default": 98
             },
        "deleted": false,
        "lastUpdatedTime": "2017-12-29T00:00:00Z"
    }
  ],
  "playlists": [
```

```
    {
      "id": "1233455555",
      "names": [
        {
          "language": "en",
          "value": "Best Hits of the 90s"
        }
      ],
      "locales": [
        {
          "country": "US",
          "language": "en"
        }
      ],
      "deleted": false,
      "lastUpdatedTime": "2017-12-29T00:00:00Z"
    }
  ]
}
```


6.6 Update API Request Parameters

Field	Description	Type	Required
HEADER: Authorization	bearer {consumerToken}	String	Yes
HEADER: Content-Type	application/json	String	Yes
POST: id	user id from Alexa	String	Yes
BODY: syncToken	newly generated sync token for this update	String	Yes
BODY: previousSyncToken	sync token for the previous update	String	Yes
BODY: stations	see S3 File above	Array	No
BODY: playlists	see S3 File above	Array	No

7 Self Healing

7.1 Prerequisites

You must implement the SyncUserContent API in your Skill Lambda.

7.2 Steps

1. At most once a day for each active user (tied to user interaction), Alexa would call the SyncUserContent API and pass in the correlationToken, accessToken (provider generated), amazon user id, previousSyncToken (latest syncToken Alexa knows for the user). If the user did not interact with the skill in 24 hours, no SyncUserContent request will be sent
2. Skill responds with one of the three statuses below:
    1. ASYNC_UPDATE – Provider detected that Alexa doesn't have the latest syncToken for the user, so, expect an asynchronous full update. Because this is a full update, upload the full personal catalog JSON to S3 as per the Account Linking scenario
    2. NO_UPDATE - Alexa & Provider are in sync for this user
    3. NO_CONTENT – Provider previousSyncToken in the self heal request is not latest and provider finds there is no personal content anymore for the user. This triggers Alexa to delete all previous content
3. Publish sync by calling the API https://{apiEndpoint}/v1/music/users/{userId}/catalog/sync and identifies user using userId in the update
    1. If response was ASYNC_UPDATE, Alexa will wait for an update (with a 300 second timeout)
    2. If the timeout expires, then Alexa would retry the request by invoking SyncUserContent with a new correlationToken; Alexa will retry after update timeouts two times maximum

7.3 SyncUserContent API Request

```
{
  "header": {
    "messageId": "1bd5d003-31b9-476f-ad03-71d471922820",
    "name": "SyncUserContent",
    "namespace": "Alexa.Audio",
    "interfaceVersion": "1.0",
    "correlationToken": "TreGEchefasWAVususuwAsTeP8"
  },
  "payload": {
    "accessToken": "e72e16c7e42f292c6912e7710c838347ae178b4a",
    "user": {
      "id": "amzn1.ask.account.ABCDEFGHIJKLMNOPQRSTUVWXYZ09123456789"
    },
    "previousSyncToken": "48freJuPaswA5UmutuCa"
  }
```

```
}
```

7.4 SyncUserContent API Response

```
{
   "header": {
      "messageId": "5f8a426e-01e4-4cc9-8b79-65f8bd0fd8a4",
      "name": "SyncUserContent",
      "namespace": "Alexa.Audio",
      "interfaceVersion": "1.0",
      "correlationToken": "TreGEchefasWAVususuwAsTeP8"
   },
   "payload": {
      "syncStatus": "ASYNC_UPDATE"
   }
}
```

7.5 Payload Details

Field	Description	Type	Required
syncStatus	ASYNC_UPDATE - Provider detected that Alexa doesn't have the latest syncToken for the user, so, expect an asynchronous full update NO_UPDATE - Alexa & Provider are in sync for this user	String	Yes

8 Error Reporting

8.1 Prerequisites

Provider implemented the ReportError API in their regular Skill Lambda.

8.2 Steps

1. Alexa while asynchronously processing the update detected error in the update
2. Alexa invokes the ReportError API of the skill to report the error. The payload includes an errorCode and descriptive errorMessage

8.3 Error Request

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

```
        "messageId": "1bd5d003-31b9-476f-ad03-71d471922820",
        "name": "ReportError",
        "namespace": "Alexa.Audio",
        "interfaceVersion": "1.0"
    },
    "payload": {
        "accessToken": "e72e16c7e42f292c6912e7710c838347ae178b4a",
        "errorCode": "InvalidSyncToken",
        "errorMessage": "Invalid syncToken found is: <> for entityId: <>. <Reason>"
    }
}
```


Note: no response from your Lambda is expected for this request.

8.4 Error Codes

Error Code	Description
InvalidContentURL	Invalid URL. Update is not processed
InvalidJSONSchema	Invalid JSON. Update is not processed
InvalidSyncStatus	Invalid syncStatus response in SyncUserContent call. Expected values: ASYNC_UPDATE or NO_UPDATE
InvalidSyncToken	Invalid usage of sync tokens. Update is not processed: * syncToken - sync token for the current update * previousSyncToken - syncToken of the previous update Rules * syncToken & previousSyncToken are mandatory attributes * syncToken cannot be null * when previousSyncToken is null, it will be treated as a full catalog update * syncToken != previousSyncToken
NoEntitiesFound	No entities found in the update. Update is not processed
UnknownCorrelationToken	Unknown or old correlationToken. Update is not processed: correlationToken is generated only by Alexa for re-sync or self-heal requests to Provider (you) to be able to correlate this request with an asynchronous update. If an async update arrives late (i.e., after Alexa sent a retry request), then the update with old correlationToken will be dropped.
UnknownEntityId	Unknown entityId used with 'deleted = true' flag. Failed entities are not processed

9 Skill Disablement

9.1 Prerequisites

Register your skill to receive SkillDisabled event by following instructions here.

9.2 Steps

1. User disables the skill
2. Alexa delivers the SkillDisabled event via the skill event lambda

10 Best Practices

1. Keep your SLA at maximum for all Async updates to within 5 minutes if not less
2. If an existing customer association is made on a customer, and a new SkillAccountLinked event occurs, ensure to remove previous customer associations
3. Ensure deletions are properly tracked and syncronized including for both customer updates and self-heals
4. To ensure the best possible CX you must implement initial sync, customer updates and self-heal
5. Provider should send playlist updates if and only if these playlists contain 1 or more tracks. Since Alexa does not keep track of the associated tracks, empty playlists will result in empty airtime for the customer.

11 Benefits of moving to V3 Schema from V2 Schema

1. Same schemas for the general catalogs so less engineering effort needed from providers.
2. Lesser number of fields in the V3 schema allow cleaner updates to be sent.

12 Playlist Schema V2

The following attributes make up a Playlist entity. The MSK Playlist entity maps to the following Alexa Schema Ontology object: **MusicPlayList**

Attribute	Type	Searchable	Returnable	Required
entityId	string	N	Y	Y
name	[{ language, name }]	Y	Y	Y
alternateName	[{ language, alias [ ] }]	Y	N	N
languageOfContent	[string]	Y	Y	N
ownerId	string	Y	Y	N
public	boolean	Y	Y	N
popularity	integer	N	Y	Y
overrides	object	N	N	N
explicitContent	boolean	Y	Y	N
deleted	boolean	Y	N	N
startDate	date	Y	N	N
endDate	date	Y	Y	N
lastUpdatedTime	timestamp	N	N	Y

13 Station Schema V2

The following attributes make up a Station entity. The MSK Station entity maps to the following Alexa Schema Ontology object: BroadcastChannel

Attribute	Type	Searchable	Returnable	Required
entityId	string	N	Y	Y
name	[{ language, name }]	Y	Y	Y
alternateName	[{ language, alias [ ] }]	Y	N	N
languageOfContent	[string]	Y	Y	N
channelId	integer	Y	Y	N
callSign	string	Y	Y	N
frequency	string	Y	Y	N
stationType	string	Y	Y	Y
ownedAndOperated	string	Y	Y	N
popularity	integer	N	Y	Y
overrides	object	N	N	N
explicitContent	boolean	Y	Y	N
deleted	boolean	Y	N	N
startDate	date	Y	N	N
endDate	date	Y	Y	N
lastUpdatedTime	timestamp	N	N	Y

14 Playlist Schema V3

The following attributes make up a Playlist entity. The MSK Playlist entity maps to the following Alexa Schema Ontology object: MusicPlayList

Attribute	Type	Searchable	Returnable	Required
id	string	N	Y	Y
names	[{ language, value }]	Y	Y	Y
alternateNames	[{ language, values [ ] }]	Y	N	N
locales	[{ country, language }]	Y	Y	Y
ownerId	string	Y	Y	N
public	boolean	Y	Y	N
popularity	{default}	N	Y	N
overrides	object	N	N	N
deleted	boolean	Y	N	Y
lastUpdatedTime	timestamp	N	N	Y

15 Station Schema V3

The following attributes make up a Station entity. The MSK Station entity maps to the following Alexa Schema Ontology object: BroadcastChannel

Attribute	Type	Searchable	Returnable	Required
id	string	N	Y	Y
names	[{ language, value }]	Y	Y	Y
alternateNames	[{ language, values [ ] }]	Y	N	N
locales	[{ country, language }]	Y	Y	Y
channelId	integer	Y	Y	N
callSign	string	Y	Y	N
frequency	string	Y	Y	N

type	string	Y	Y	Y
popularity	{default}	N	Y	N
overrides	object	N	N	N
deleted	boolean	Y	N	Y
lastUpdatedTime	timestamp	N	N	Y

16 Mappings between the V2 and V3 Fields

Field Names	Changes	V2	V3	Datatype (V3)
EntityId	entityId has been renamed to id	entityId	id	String
Name	name has been renamed to names	name	names	Array of Object {language(String), values(String)}
AlternateName	alternateName has been renamed to alternateNames	alternateName	alternateNames	Array of Object {language(string), array of values(String)}
languageOfContent	languageOfContent has been removed in V3 schemas	languageOfContent	N/A	Not in V3
Locales	Locales has been introduced in V3 schemas for more granular locale and country assignment	N/A	locales	Array of {country(String), language (String)}
popularity	popularity has been changed from an int to an object of default(int)	popularity	popularity	Object of default(int)
explicitContent	explicitContent has been removed in V3	explicitContent	N/A	explicitContent has been removed in V3