Spotify Audiobook ONIX Specification

Provider Reference

SPOTIFY

AUGUST 26, 2026

Important Legal Notice — Please Read Before Proceeding

This information and all other documentation (in printed or electronic form) in connection herewith (the "Documentation") are provided for reference purposes only. While efforts were made to verify the completeness and accuracy of the Documentation, it is provided "as is" without any warranty whatsoever and to the maximum extent permitted, Spotify disclaims all warranties, including without limitation the implied warranties of merchantability, non-infringement and fitness for a particular purpose, with respect to the Documentation. Spotify shall not be responsible for any damages, including without limitation, direct, indirect, consequential or incidental damages, arising out of the use of, or otherwise related to, the Documentation. Notwithstanding anything to the contrary, nothing contained in the Documentation is intended to, nor shall have the effect of, creating any warranties or representations from Spotify (or its suppliers or licensors).

The Documentation is strictly confidential and shall not be disclosed to any unauthorized person nor to any third party. The Documentation shall be used only for the intended business purpose for which it was disclosed.

Spotify shall remain the owner of any and all rights concerning the Documentation, including but not limited to intellectual and industrial property rights, trade secret rights, usage rights and the right to register trademarks or other intellectual property rights, and no transfer of any ownership, license, usage or other rights in relation to the Documentation or any information, documents and materials derived therefrom is granted unless established under a separate written agreement.

By accessing the Documentation, you confirm that you have read and agreed to the foregoing legal notice on behalf of yourself and the organization you are representing.

Table of Contents

1. Introduction

Document Overview

This specification describes how to deliver ***audiobooks*** to Spotify using ONIX 3.0+ metadata. It covers which fields Spotify reads, how they're interpreted, and what you need to get an ***audiobook*** live on the platform.

The intended audience is publishers and distributors delivering ***audiobook*** metadata to Spotify. A working familiarity with ONIX 3.0+ and XML is recommended for users of this documentation.

How This Specification Is Organized

The specification is structured in two layers:

Essentials (Sections 1-3) walks you through delivering your first ***audiobook*** from scratch. It covers what a delivery is, what each piece of the ONIX does, and a complete worked example. If you're new to delivering ***audiobooks*** to Spotify, start here.

Reference (Sections 4+) provides in-depth documentation for every ONIX element Spotify reads and breakdowns of delivery patterns.

For clarity, the Essentials articles describe the high-level concepts and provide enough information to understand how to send Spotify a ‘basic’ ***audiobook*** delivery, whereas the Reference articles provide a more comprehensive understanding of the full capabilities and logic of Spotify’s ingestion processes.

How to Read Reference Articles

Each reference article explains the field’s purpose and usage, then covers:

- **Spotify Behavior**: Describes how Spotify selects, processes, and interprets the field value.
- **Fallback Logic**: When applicable, lists the alternative sources Spotify checks, in priority order, when the primary value is missing, empty, or invalid.
- **Example**: Provides an XML example snippet of a valid delivery and shows the resulting Spotify behavior.

Defined Document Terms

Where applicable, this specification uses bold terms, italicized keywords to indicate a field’s requirement level:

- ***End User***: The user of the Spotify application.
- ***Audiobook***: The individual listening experience consumed by an ***end user***
- ***Lifecycle States***: Milestones that an ***Audiobook*** goes through in the Spotify ***end user*** experience
    - ***Ingestion***: The milestone referring to an ***audiobook*** that has entered the Spotify ecosystem.
    - ***Countdown Page***: The milestone referring to an ***audiobook*** that has a detailed page in the Spotify application before release.
    - ***Full Release***: The milestone referring to an ***audiobook*** that has a detailed page in the Spotify application where ***end users*** are able to consume audio.
- ***Inactive***: No new ***end users*** may discover or purchase your audiobook. In some contexts, an ***end user*** who previously engaged with your audiobook may continue to access it, such as continuing access for **à la carte** purchases. See your contract for additional details.
    - Additionally, a ***takedown*** makes an ***audiobook*** ***inactive.***
- ***Must***: Required for the ***audiobook*** to become available to ***end users*** on Spotify.
- ***Should***: Recommended to provide the best ***end user*** experience.

2. Key Concepts

This section explains how ***audiobook*** delivery to Spotify works at a high level: what you send, what the ONIX controls, and how an ***audiobook*** goes from a file on your SFTP to an ***audiobook*** that ***end users*** can find and play in the Spotify application.

What is an Audiobook Delivery?

An ***audiobook*** on Spotify is built from three things delivered together:

COMPONENT	WHAT IT IS	FORMAT
ONIX metadata	An XML file describing the ***audiobook***. This includes things like its title, contributors, description, pricing, availability, and more	ONIX 3.0+ (one `<Product>` per file recommended)
Audio files	The ***audiobook*** content, one file per chapter, in sequential order	FLAC (preferred), WAV, or MP3
Cover art	The image shown to ***end users*** in the Spotify app	PNG (preferred), JPEG, GIF, or TIFF 1:1 aspect ratio minimum 640px

In addition to the basics for a delivery, it is recommended to include a small audio clip to be used as a spoiler-free preview and, where applicable, a supplemental PDF. More detail about the supported components, filenaming patterns, and delivery structure can be found in Feed Structure.

All files ***must*** be delivered to a Spotify-provided SFTP location, and ***should*** be organized in a directory named by the ***audiobook's*** GTIN-13. Well-formed deliveries are typically reflected on Spotify within 7 days of upload.

What does ONIX Control?

An ONIX delivery to Spotify carries three distinct categories of data: **feed metadata**, **content metadata**, and **commercial metadata**.

Feed Metadata

Feed metadata tells Spotify how to process the ONIX delivery. It answers key questions such as who sent the file, when it was sent, what type of product it is, and how Spotify should identify it.

- **Sender:** identifies the organization that sent the file. Spotify matches this to your provider account. Use the same **<SenderName>** consistently across deliveries.
- **Sent Date Time:** a timestamp on each delivery that protects against out-of-order processing. If Spotify has already ingested a newer file for the same ***audiobook***, the older delivery is skipped.
- **Product Identifiers:** the GTIN-13 that ties the ONIX file to the audio and cover art on the SFTP. At least one identifier must be provided**Product Form:** confirms the delivery is an audio product via **<ProductForm>**
- **Product Content Type:** confirms the content is an ***audiobook*** via **<PrimaryContentType>**
- **Notification Type:** tells Spotify what to do with the file. `03` is a normal delivery, `05` is a ***takedown***, and `04` is a partial update (commercial metadata changes only). This also determines whether a **<ProductSupply>** block is required in the same delivery.

Content Metadata

Content metadata describes the ***audiobook*** itself. Spotify uses this data for discovery and recommendations.

- **Title:** the display name, determined by **<TitleDetail>** and **<Subtitle>**
- **Description:** the summary, determined by **<TextContent>**
- **Contributors:** authors, narrators, and translators, determined by **<Contributor>** with role codes like `A01` (author) and `E07` (narrator)
- **Subjects:** Classifications used for categorization, determined by **<Subject>** such as BISAC, Thema, or CLIL
- **Edition:** Unabridged, Abridged, Adapted, etc., determined by **<EditionType>**
- **Series / Collection:** groups related audiobooks and shows position (e.g., "Harry Potter #3"), determined by **<Collection>**
- **Publisher:** the organization that published the audiobook, determined by **<Publisher>** with **<PublishingRole>** `01`
- **Duration:** the ONIX-reported runtime, determined by **<Extent>** with type `09`.
- **Copyright:** copyright and phonogram ownership, determined by **<CopyrightStatement>**
- **Chapter-Level Metadata:** chapter names displayed in the listening experience, determined by **<ContentItem>** composites in **<ContentDetail>**

Each of these are covered in their own reference article in the Content Metadata section.

Spotify utilizes your ***audiobook*** metadata as the primary source for how your ***audiobook*** appears to ***end users.*** During processing, Spotify may apply minor cosmetic adjustments (such as removing HTML, applying spelling corrections, trimming whitespace, etc) to ensure a consistent ***end user*** experience

Commercial Metadata

Commercial metadata controls where, when, and how the ***audiobook*** is available to ***end users***. It answers four questions, each building on the previous:

- **Audiobook Availability:** is the ***audiobook*** active or taken down? When does it go on sale? Determined by `<NotificationType>`, `<PublishingStatus>`, `<MarketPublishingStatus>`, and `<PublishingDate>`.
- **Market Availability:** which countries is it available in? Determined by `<SalesRights>`, `<ROWSalesRightsType>`, and the `<Territory>` within each `<Market>`.
- **Price Acceptance:** which `<Price>` composites does Spotify accept when determining the suggested retail price? Determined by `<PriceType>`, `<PriceAmount>`, `<CurrencyCode>`, and `<PriceDate>`.
- **Price Selection:** of the accepted prices, which suggested retail price applies in each market? Determined by the `<Territory>` on each `<Price>` and price ranking rules.

These four areas work together. A price only applies in markets the ***audiobook*** is available in, and market availability only matters if the ***audiobook*** is ***active***. Getting the correct outcome with the delivery means making sure all four agree.

The market availability expressed in the ***audiobook's*** ONIX metadata tells Spotify where you intend the ***audiobook*** to be offered. We will always honor ONIX territory rights, and actual ***end user*** availability may be further constrained based on Spotify's discretion and the terms of your distribution agreement.

Each of these categories is covered in its own reference article in the Commercial Metadata section. These articles also detail the fine-grained pricing and availability controls that Spotify offers.

Where are Key Concepts in ONIX?

The table below maps each top-level ONIX block to the category of data it covers. For a complete example showing every block in context, see Getting Started: Delivering Your First Audiobook.

ONIX BLOCK	CATEGORY	KEY CONCEPT
`<Header>`	Feed	Sender, Sent Date Time
`<Product>` (top level)	Feed	Notification Type, Product Identifiers, Product Form, Product Content Type
`<DescriptiveDetail>`	Content	Title, Contributors, Subjects, Edition, Series / Collection, Duration
`<CollateralDetail>`	Content	Description

ONIX BLOCK	CATEGORY	KEY CONCEPT
`<ContentDetail>`	Content	Chapter-Level Metadata
`<PublishingDetail>`	Content + Commercial	Publisher, Copyright, Publishing Status, Sale Start Date, Sales Rights
`<ProductSupply>`	Commercial	Territory, Pricing

What is the Audiobook Lifecycle?

Once uploaded, an ***audiobook*** progresses through three milestones on its way to ***end users: Ingestion, Countdown Page, and Full Release***. Each milestone builds on the previous. For example, everything required for a ***Countdown Page*** is also required for a ***Full Release***.

Ingestion

The ***audiobook*** enters Spotify's catalog. It appears in your Spotify for Authors account but is not visible to ***end users***.

REQUIREMENT	SOURCE
GTIN-13 or ISBN-13	ONIX **`<ProductIdentifier>`**
Identification of sender	ONIX **`<Header>`** / **`<Sender>`**
Sent date/time	ONIX **`<Header>`** / **`<SentDateTime>`**
Audiobook format confirmed	ONIX **`<ProductForm>`** (must be an accepted audio form, e.g. `AJ`)
Notification type	ONIX **`<NotificationType>`** (e.g. `03` for a normal delivery)
Product availability	ONIX **`<ProductSupply>`** / **`<SupplyDetail>`** / **`<ProductAvailability>`**
Price or unpriced item type	ONIX **`<Price>`** or **`<UnpricedItemType>`**

Countdown Page

The ***audiobook*** populates a detail page in the Spotify application, containing a Countdown Page, Spotify's pre-release experience. The following are also required to reach this milestone:

REQUIREMENT	SOURCE
Audiobook title	ONIX **`<TitleDetail>`**
Audiobook description	ONIX **`<TextContent>`**
Author (at least one)	ONIX **`<Contributor>`** with an author role (e.g. `A01`)
Name of publisher	ONIX **`<Publisher>`** with **`<PublishingRole>`** `01`
Cover art	Image file delivered via SFTP

Note that a ***Countdown Page*** is limited to ***audiobooks*** with a future sale start date. The sale start date is set by **`<PublishingDate>`** with role `01`. If that date is still in the future, the ***audiobook*** stays on its ***Countdown Page*** until the date arrives.

Full Release

The ***audiobook*** is fully available for listening; ***end users*** can access the audio content via their **Audiobooks in Premium (ABP)** subscription and purchase it via **à la carte**.

This only occurs after the sale start date has been reached. Redelivery is not required to switch from ***Countdown Page*** to ***Full Release***.

The following are also required to reach this milestone:

REQUIREMENT	SOURCE
Narrator (at least one)	ONIX **`<Contributor>`** with a narrator role (e.g. `E07`)
Audio files	Audio files delivered via SFTP

Updates and Takedowns

Updates to assets and/or metadata are supported:

- For metadata updates, redeliver the complete ONIX file.
  - Spotify does not support partial updates via ONIX block updates.
- The most recent file(s) received for a given GTIN completely replace the previous version(s).
- When updating audio, always re-send all audio files; delivering a single file may replace the entire ***audiobook's*** content with that one file.

Takedowns make an ***audiobook inactive:***

- Triggered by either
  - **<PublishingStatus>** `11`
  - **<NotificationType>** `05`.
- There is no way to schedule a ***takedown*** for a future date.
- The ***audiobook*** becomes ***inactive*** the moment Spotify has ingested the ONIX file.

Recommended for the Best Experience

These fields are not required to reach either the ***Countdown Page*** or ***Full Release*** milestones, but significantly improve the ***end user*** experience and are strongly recommended to include:

FEATURES	WHY IT MATTERS
Chapter-Level Metadata	Without it, chapters display as "Track 1," "Track 2," etc.
Subjects	Used for categorization and recommendations
Edition	Helps ***end users*** distinguish abridged from unabridged
Series / Collection	Groups related ***audiobooks*** and shows position in the series
Copyright	Displays copyright information to ***end users***
Sales Rights	Determines which countries the ***audiobook*** is available in

3. Getting Started: Delivering Your First Audiobook

This is a complete walkthrough of an ***audiobook*** delivery. It covers an accepted ONIX file, the associated files on SFTP, and what Spotify does with each.

We'll use a single, simple example ***audiobook*** delivered to the US, GB, CA, and AU markets. Variances and nuances are available in the relevant reference articles.

The full ONIX and SFTP delivery is broken down section by section below. To see the complete, uninterrupted ONIX file, refer to the Example Audiobook.

What Gets Delivered

For our example ***audiobook***, `9781234567890`, files are uploaded to the SFTP with the following directory structure:

```
9781234567890/
├── 9781234567890.xml                        (ONIX metadata)
├── 9781234567890.png                        (Cover art)
└── 9781234567890.zip
    ├── 9781234567890_0001_OF_0003.flac      (Track 1: Introduction)
    ├── 9781234567890_0002_OF_0003.flac      (Track 2: Chapter 1)
    └── 9781234567890_0003_OF_0003.flac      (Track 3: Closing Credit)
```

The GTIN serves as the shared identifier across all files in the delivery. The ONIX metadata, audio, and cover art are all matched to the same ***audiobook*** by this value.

Feed Metadata: Identification

Identifying the Provider

Every ONIX file starts with a **`<Header>`** that tells Spotify who sent the file and when.

```
<ONIXMessage release="3.0" xmlns="http://ns.editeur.org/onix/3.0/reference">
 <Header>
   <Sender>
     <SenderIdentifier>
       <SenderIDType>01</SenderIDType>
       <IDValue>example-publisher</IDValue>
     </SenderIdentifier>
     <SenderName>Example Publisher Inc.</SenderName>
```

```
   </Sender>
   <SentDateTime>20250615T120000</SentDateTime>
 </Header>
<!-- ... -->
</ONIXMessage>
```

ELEMENT	WHAT IT DOES
<SenderName>	Identifies who sent the file. Use the same value consistently across deliveries.
<SentDateTime>	Protects against out-of-order processing. If Spotify has already processed a newer file for this ***audiobook***, this delivery is skipped. Each delivery should have a unique timestamp.

Identifying Which Audiobook

The first elements inside **<Product>** confirm that the delivery is an ***audiobook***, which is crucial because Spotify will reject eBook or Print metadata. These elements also include key unique identifiers (e.g. ISBN or GTIN) for consistent identification of the ***audiobook*** that this delivery represents.

```
<ONIXMessage>
 <!-- <Header> ... </Header> -->
 <Product>
   <RecordReference>9781234567890</RecordReference>
   <NotificationType>03</NotificationType>
   <ProductIdentifier>
     <ProductIDType>15</ProductIDType>
     <IDValue>9781234567890</IDValue>
   </ProductIdentifier>
   <ProductIdentifier>
     <ProductIDType>03</ProductIDType>
     <IDValue>9781234567890</IDValue>
   </ProductIdentifier>
   <!-- ... -->
 </Product>
</ONIXMessage>
```

ELEMENT	WHAT IT DOES
<NotificationType> `03`	Tells Spotify this is a standard, complete, and active delivery.
<ProductIDType> `03`	GTIN-13. This is Spotify's primary identifier for the ***audiobook***. It links the ONIX to the audio files and cover art on SFTP.

ELEMENT	WHAT IT DOES
<ProductIDType> 15	ISBN-13. This is a secondary identifier for the ***audiobook***, indicating that the GTIN is also an ISBN.

Confirming This Is an Audiobook

Spotify needs to verify the delivery is actually an ***audiobook*** before processing it.

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
   <!-- ... -->
   <DescriptiveDetail>
     <ProductComposition>00</ProductComposition>
     <ProductForm>AJ</ProductForm>
     <!-- ... -->
   </DescriptiveDetail>
   <!-- ... -->
 </Product>
</ONIXMessage>
```

ELEMENT	WHAT IT DOES
<ProductForm> AJ	Confirms this is a downloadable audio file.

Content Metadata: Describing the Contents

The Content Metadata is carried by the rest of the **<DescriptiveDetail>**, along with **<CollateralDetail>** and **<ContentDetail>**.

Title and Subtitle

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
   <!-- ... -->
   <DescriptiveDetail>
     <!-- ... -->
     <TitleDetail>
       <TitleType>01</TitleType>
       <TitleElement>
         <TitleElementLevel>01</TitleElementLevel>
         <TitleText language="eng">The Audiobook Example</TitleText>
         <Subtitle language="eng">An Example of ONIX</Subtitle>
```

```
      </TitleElement>
    </TitleDetail>
    <!-- ... -->
  </DescriptiveDetail>
  <!-- ... -->
 </Product>
</ONIXMessage>
```

Spotify selects the **`<TitleDetail>`** with **`<TitleType>`** `01` and reads the product-level **`<TitleElement>`** (**`<TitleElementLevel>`** `01`). On most surfaces, ***end users*** see the combined form: **"The Audiobook Example: An Example of ONIX"**.

Description

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
   <!-- ... -->
   <CollateralDetail>
     <TextContent>
       <TextType>03</TextType>
       <ContentAudience>00</ContentAudience>
       <Text language="eng">An Example description that would be displayed.</Text>
     </TextContent>
   </CollateralDetail>
   <!-- ... -->
 </Product>
</ONIXMessage>
```

Spotify selects the **`<TextContent>`** with **`<TextType>`** `03` description whose language matches the ***audiobook*** language determined according to Common ONIX Processing Rules : Language Resolution. This is the summary shown on the ***audiobook's*** detail page.

Contributors

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
   <!-- ... -->
   <DescriptiveDetail>
     <!-- ... -->
     <Contributor>
       <SequenceNumber>1</SequenceNumber>
       <ContributorRole>A01</ContributorRole>
       <PersonName>John Smith</PersonName>
     </Contributor>
     <Contributor>
```

```
      <SequenceNumber>2</SequenceNumber>
      <ContributorRole>E07</ContributorRole>
      <PersonName>Jane Doe</PersonName>
    </Contributor>
    <Contributor>
      <SequenceNumber>3</SequenceNumber>
      <ContributorRole>B06</ContributorRole>
      <PersonName>Alex Johnson</PersonName>
    </Contributor>
    <!-- ... -->
  </DescriptiveDetail>
  <!-- ... -->
 </Product>
</ONIXMessage>
```

At least one author and one narrator are required.

Spotify classifies contributors by their **`<ContributorRole>`**:

ROLE CODE	CATEGORY	THIS EXAMPLE
`A01`	Author	John Smith
`E07`	Narrator	Jane Doe
`B06`	Translator	Alex Johnson

Subject Codes

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
  <!-- ... -->
  <DescriptiveDetail>
    <!-- ... -->
    <Subject>
      <SubjectSchemeIdentifier>10</SubjectSchemeIdentifier>
      <SubjectCode>FIC028000</SubjectCode>
    </Subject>
    <Subject>
      <SubjectSchemeIdentifier>93</SubjectSchemeIdentifier>
      <SubjectCode>FL</SubjectCode>
    </Subject>
    <!-- ... -->
  </DescriptiveDetail>
  <!-- ... -->
 </Product>
</ONIXMessage>
```

Subject codes help Spotify categorize the ***audiobook*** for discovery and recommendations. Scheme `10` is BISAC, `93` is Thema.

Spotify considers your subject codes when evaluating ***audiobooks*** for explicit classification. For the best ***end user*** experience please provide subject codes that accurately reflect the contents of your ***audiobook***.

Series, Edition, and Duration

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
   <!-- ... -->
   <DescriptiveDetail>
     <!-- ... -->
     <Collection>
       <CollectionType>10</CollectionType>
       <TitleDetail>
         <TitleType>01</TitleType>
         <TitleElement>
           <TitleElementLevel>02</TitleElementLevel>
           <PartNumber>1</PartNumber>
           <TitleText>The Example Series</TitleText>
         </TitleElement>
       </TitleDetail>
     </Collection>
     <EditionType>UBR</EditionType>
     <Extent>
       <ExtentType>09</ExtentType>
       <ExtentValue>10680</ExtentValue>
       <ExtentUnit>06</ExtentUnit>
     </Extent>
   </DescriptiveDetail>
   <!-- ... -->
 </Product>
</ONIXMessage>
```

ELEMENT	RESULT
`<Collection>` with **`<CollectionType>`** `10`	Groups this ***audiobook*** into "The Example Series #1"
`<EditionType>` `UBR`	Marks it as Unabridged
`<Extent>` with type `09`, unit `06`	Reports the ONIX duration as 10,680 seconds.

Publisher and Copyright

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
   <!-- ... -->
   <PublishingDetail>
     <Publisher>
       <PublishingRole>01</PublishingRole>
       <PublisherName>Example Publisher Inc.</PublisherName>
     </Publisher>
     <CopyrightStatement>
       <CopyrightType>C</CopyrightType>
       <CopyrightYear>2025</CopyrightYear>
       <CopyrightOwner>
         <PersonName>John Smith</PersonName>
       </CopyrightOwner>
     </CopyrightStatement>
     <CopyrightStatement>
       <CopyrightType>P</CopyrightType>
       <CopyrightYear>2025</CopyrightYear>
       <CopyrightOwner>
         <CorporateName>Example Publisher Inc.</CorporateName>
       </CopyrightOwner>
     </CopyrightStatement>
     <!-- ... -->
   </PublishingDetail>
   <!-- ... -->
 </Product>
</ONIXMessage>
```

The publisher with **`<PublishingRole>`** `01` is used as the display publisher. Both copyright statements are used, the text copyright (C) and the phonogram/sound recording copyright (P).

Chapter-Level Metadata

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
   <!-- ... -->
   <ContentDetail>
     <ContentItem>
       <LevelSequenceNumber>1</LevelSequenceNumber>
       <TitleDetail>
         <TitleType>01</TitleType>
         <TitleElement>
           <TitleElementLevel>01</TitleElementLevel>
           <TitleText>Introduction</TitleText>
         </TitleElement>
       </TitleDetail>
    <!-- ... -->
     </ContentItem>
     <ContentItem>
       <LevelSequenceNumber>2</LevelSequenceNumber>
         <TitleText>Chapter 1: The Example Chapter Name</TitleText>
```

```
      <!-- ... -->
    </ContentItem>
    <ContentItem>
      <LevelSequenceNumber>3</LevelSequenceNumber>
        <TitleText>Closing Credit</TitleText>
      <!-- ... -->
    </ContentItem>
  </ContentDetail>
  <!-- ... -->
 </Product>
</ONIXMessage>
```

Each **`<ContentItem>`** provides the display name for one audio track. The **`<LevelSequenceNumber>`** determines the order. Without chapter-level metadata, Spotify defaults track names to "Track 1," "Track 2," etc.

There ***must*** be exactly one **`<ContentItem>`** per non-sample audio file.

Commercial Metadata: Where, When, and How Much

The **`<PublishingDetail>`** and **`<ProductSupply>`** blocks control whether the ***audiobook*** is available, in which countries, and at what price.

Audiobook Availability: Is It Active? When Does It Go on Sale?

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
   <!-- ... -->
   <PublishingDetail>
     <!-- ... -->
     <PublishingStatus>04</PublishingStatus>
     <PublishingDate>
       <PublishingDateRole>01</PublishingDateRole>
       <Date>20250115</Date>
     </PublishingDate>
     <!-- ... -->
   </PublishingDetail>
   <!-- ... -->
 </Product>
</ONIXMessage>
```

ELEMENT	WHAT IT DOES
`<PublishingStatus>` `04`	The ***audiobook*** is **active**.

ELEMENT	WHAT IT DOES
`<PublishingDate>` role `01`	Sets the **sale start date** to January 15, 2025. Before this date, the ***audiobook*** has a ***Countdown Page***. After this date, it reaches ***Full Release***.

Market Availability: Which Countries?

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
   <!-- ... -->
   <PublishingDetail>
     <!-- ... -->
     <SalesRights>
       <SalesRightsType>01</SalesRightsType>
       <Territory>
         <CountriesIncluded>US GB CA AU</CountriesIncluded>
       </Territory>
     </SalesRights>
   </PublishingDetail>
   <!-- ... -->
 </Product>
</ONIXMessage>
```

`<SalesRights>` with Type `01` declares the master set of countries where the title may be sold. The **`<Market>`** territory inside **`<ProductSupply>`** is then intersected with this set, and may further restrict availability beyond what **`<SalesRights>`** allows.

Pricing and Supply: How Much?

```
<ONIXMessage>
 <!-- <Header>...</Header> -->
 <Product>
   <!-- ... -->
   <ProductSupply>
     <Market>
       <Territory>
         <CountriesIncluded>US GB CA AU</CountriesIncluded>
       </Territory>
     </Market>
     <MarketPublishingDetail>
       <MarketPublishingStatus>04</MarketPublishingStatus>
     </MarketPublishingDetail>
     <SupplyDetail>
       <Supplier>
         <SupplierRole>00</SupplierRole>
         <SupplierName>Example Supplier</SupplierName>
       </Supplier>
```

```
        <ProductAvailability>20</ProductAvailability>
        <Price>
          <PriceType>01</PriceType>
          <PriceAmount>24.99</PriceAmount>
          <CurrencyCode>USD</CurrencyCode>
          <Territory>
            <CountriesIncluded>US</CountriesIncluded>
          </Territory>
        </Price>
        <Price>
          <PriceType>01</PriceType>
          <PriceAmount>19.99</PriceAmount>
          <CurrencyCode>GBP</CurrencyCode>
          <Territory>
            <CountriesIncluded>GB</CountriesIncluded>
          </Territory>
        </Price>
      </SupplyDetail>
    </ProductSupply>
  </Product>
</ONIXMessage>
```

Here's how each piece works:

ELEMENT	WHAT IT DOES
`<Market>` / **`<Territory>`**	This supply covers US, GB, CA, and AU, matching the sales rights.
`<MarketPublishingStatus>` `04`	Confirms the ***audiobook*** is active in these markets.
`<ProductAvailability>` `20`	The product is available.
`<Price>` with **`<PriceType>`** `01`	The suggested retail price (RRP excluding tax). The **`<Territory>`** on each price restricts it to specific markets. $24.99 USD applies to the US £19.99 GBP applies to GB. There are no prices for other markets, so the ***audiobook*** will not be sold in CA and AU. It is acceptable to send a price without territory restrictions. See price selection logic for more details.

How Spotify Interprets This Audiobook

When Spotify processes this delivery, the audiobook is live in two countries, available via both **Audiobooks in Premium** and **à la carte**, with chapter names displayed in the listening experience.

ELEMENT	RESULT
Title	The Audiobook Example: An Example of ONIX

ELEMENT	RESULT
Author	John Smith
Narrator	Jane Doe
Description	An Example description that would be displayed. Translated by: Alex Johnson
Publisher	Example Publisher Inc.
Edition	Unabridged
Series	The Example Series #1
Subjects	BISAC FIC028000, Thema FL
Copyright	C 2025 John Smith, P 2025 Example Publisher Inc.
Chapters	Introduction, Chapter 1: The Example Chapter Name, Closing Credit
Status	Active
Sale start	January 15, 2025
Markets	US, GB
Price (US)	$24.99 USD
Price (GB)	£19.99 GBP

What's Next

This walkthrough covered every ONIX element in a complete basic delivery. The reference sections that follow document each field in detail, including fallback behavior, edge cases, accepted code values, and validation rules:

- **Feed Structure and File Name Patterns:** covers file naming patterns, delivery methods, identifiers, and update rules
- **Feed Metadata:** covers Sender, Sent Date Time, Product Identifiers, Product Form, Product Content Type, and Notification Type
- **Content Metadata:** covers Title, Description, Contributors, Subjects, Series / Collection, Edition, Duration, Copyright, and Chapter-Level Metadata

- **Commercial Metadata:** covers Audiobook Availability, Market Availability, Price Acceptance, and Price Selection
- **Validation:** covers what happens when fields are missing, invalid, or ambiguous

4. Feed Structure and File Name Patterns

This document describes the file formats and structures required for publishers to deliver ***audiobooks*** to Spotify by pushing to Spotify's SFTP. Please see the Feed Setup support article for operational details.

Compliance with these guidelines is mandatory. Submissions that do not meet these specifications may be rejected, requiring resubmission and potentially delaying your title's availability on our platform.

Well-formed deliveries are typically reflected on Spotify within 7 days of upload.

Identifiers

Spotify requires GTIN-13 (Global Trade Item Number) or ISBN-13 (International Standard Book Number) as the primary identifier, to be used to associate various assets and metadata with the same unique audiobook.

Throughout this document, `{GTIN}` refers to either an ISBN-13 or GTIN-13 identifier, both in the filename patterns described and in all other commentary. You may use whichever identifier your catalog supports.

Please note that we do not support changes to GTIN. If you are changing a GTIN for your catalog, you ***must*** send a ***takedown*** for the existing GTIN and deliver the new GTIN (inclusive of all supporting assets) as a fresh/new delivery. Spotify does not currently have a process for reassociating old reviews or consumption with the new GTIN.

File Structure

Deliveries ***must*** be sent to a Spotify-provided SFTP location. Spotify does not pull from external servers. Please see the Feed Setup support article for more details and FAQ.

Zip Delivery (Preferred)

Each audiobook should have all of its files stored in its own directory, named by GTIN. It is also supported to deliver files directly into the root (provided files comply with the filename specifications below), but this is not recommended due to the more challenging operator experience when reviewing ingestion issues.

If you deliver zips, at a minimum, they should contain all non-sample audio files for the ***audiobook***. You may also deliver any other asset or metadata associated with the ***audiobook*** within the same zip file.

```
Root/9781234567890/
├── 9781234567890.xml         (ONIX 3.0 Metadata File)
├── 9781234567890.png         (Cover Art Image)
├── 9781234567890.zip         (ZIP archive containing main audio files)
│   ├── 9781234567890_0001_OF_0003.flac (Audiobook Contents 1/3)
│   ├── 9781234567890_0002_OF_0003.flac (Audiobook Contents 2/3)
│   ├── 9781234567890_0003_OF_0003.flac (Audiobook Contents 3/3)
│   └── 9781234567890_sample.flac (Preview/Sample Audio File)
└──  9781234567890.pdf        (Supplemental PDF File)
```

Loose File Delivery

You may also deliver an audiobook without zipping the audio files within a GTIN-named folder, provided there is adequate signal of upload completion utilizing one (or more) of the following two methods:

1) **[Preferred]** Audiobook Contents are named as `{GTIN}_{sequence}_of_{total}.{filetype}`
2) Delivery of a `.ok` or `.complete` marker file at the end of an upload

If we do not receive the complete upload within 7 days, the audiobook files will not be ingested.

Again, it is supported (but not advised) to deliver files directly into the root instead of a GTIN-named folder.

```
Root/9781234567890/
├── 9781234567890.xml         (ONIX 3.0 Metadata File)
├── 9781234567890.png         (Cover Art Image)
├── 9781234567890_0001_OF_0003.flac (Audiobook Contents 1/3)
├── 9781234567890_0002_OF_0003.flac (Audiobook Contents 2/3)
├── 9781234567890_0003_OF_0003.flac (Audiobook Contents 3/3)
├── 9781234567890_sample.flac (Preview/Sample Audio File)
├── 9781234567890.ok        (Completion Marker File)
└── 9781234567890.pdf        (Supplemental PDF File)
```

Multi-Product ONIX Files

You *may* deliver metadata via multi-product ONIX files. However, we advise against doing so as this increases processing time, increases the risk of ingestion failures, and adds complexity to debugging and error workflows.

If you deliver metadata via multi-product ONIX files, it is strongly recommended that you name files by datetime to simplify error workflows and pipeline visibility. Deliver them to the root directory, and not in a zip file.

```
Root/
├── 20260701_143022.xml       (ONIX 3.0 Metadata File)
└── 9781234567890/
    ├── 9781234567890.png         (Cover Art Image)
    ├── 9781234567890_0001_OF_0003.flac (Audiobook Contents 1/3)
    ├── 9781234567890_0002_OF_0003.flac (Audiobook Contents 2/3)
    ├── 9781234567890_0003_OF_0003.flac (Audiobook Contents 3/3)
    ├── 9781234567890_sample.flac (Preview/Sample Audio File)
    └──  9781234567890.pdf        (Supplemental PDF File)
```

Accepted Filenaming Patterns and Filetypes

When multiple file types are allowed, Spotify's preference will be indicated in bold. Note that our filename parsing allows for substantial flexibility, expressed as wildcards (*) below, but it is strongly recommended to exclude extraneous information to avoid processing errors.

ASSET TYPE	ACCEPTED FILENAMES	ACCEPTED FILETYPES	NOTES
Book Metadata	***Delivery via single-product ONIX is strongly preferred.*** If delivering single-product ONIX: **Preferred:** `{GTIN}.{filetype}` Supported: `*.{filetype}` ***Must*** include GTIN if not in a GTIN-named folder If delivering multi-product ONIX: **Preferred:** `{datetime}.{filetype}` Supported: `*.{filetype}`	`.xml` `.onix`	Only ONIX 3.0+ is supported at this time. Spotify does not support block (partial) updates, although note that your commercial and content metadata are processed independently, which means a partial failure/success may occur.. Whenever audio is delivered, also deliver an ONIX file to enable CLM (Chapter Level Metadata) capture, if your metadata supports it.
Zip File (Optional)	**Preferred:** `{GTIN}.zip` Supported: `*_{GTIN}_*.zip` If multiple zips are provided due to filesize limitations, name as `{GTIN}_{sequence}_of_{total}.zip`	`.zip`	Note that .rar and .7z are not supported.

ASSET TYPE	ACCEPTED FILENAMES	ACCEPTED FILETYPES	NOTES
Cover Art	**Preferred:** `{GTIN}.{filetype}` Supported: `*_{GTIN}_*.{filetype}`	`.png` `.jpg/.jpeg` `.gif` `.tif/.tiff`	Resolution minimum of 640px and maximum of 10000px. Minimum of 3000px is strongly preferred, as well as a “square” (1:1 ratio) image. Failure to provide a square image may degrade the end user experience. RGB color space only (No CMYK). Only one cover image per ***audiobook*** - the most recent image file received will be used.
Audio (Audiobook Contents)	**Preferred:** `{GTIN}_{SEQUENCE}_OF_{TOTAL}.{filetype}` Supported: `{GTIN}_{SEQUENCE}.{filetype}` `{SEQUENCE}_{GTIN}.{filetype}` `{GTIN}_{SEQUENCE}_*.{filetype}` `*_{SEQUENCE}_{GTIN}.{filetype}`	`.flac` `.wav .mp3`	Individual files may be no longer than 12 hours long. Filenames should match exactly the values in the ONIX manifest, if present. Please observe that extraneous inclusion of title names, chapter names, or other information may result in ***audiobook*** audio files inadvertently being treated as sample audio (described below). Maximum of 999 tracks (including sample audio) per audiobook. Note that `.m4a`, `.m4b`, `.aac`, `.ogg`, and `.wma` are not supported.
Audio (Sample)	**Preferred:** `{GTIN}_sample.{filetype}` Supported: `*_{GTIN}_sample.{filetype}` `Sample_{GTIN}_*.{filetype}` `*_{GTIN}_pre.{filetype}` `*_{GTIN}_preview.{filetype}`	`.flac` `.wav .mp3`	Samples will be available to the public. Content ***must*** be appropriate for general audiences and spoiler-free.
Supplemental PDF (Optional)	**Preferred:** `{GTIN}.pdf` Supported**:** `*_{GTIN}_*.pdf` If multiple PDFs are provided, name as `{GTIN}_{sequence}_of_{total}.pdf`	`.pdf`	We generally recommend limiting file size to <50MB to enable a reasonable end user experience. However, Spotify systems can support larger files. PDFs are currently the only supported format for supplemental materials

Note that while our *supported* filename patterns typically include the GTIN, if a file is delivered within a GTIN-named zip file or directory, our system will attempt to make the correct ***audiobook*** association and

continue with ingestion. However, it is generally not advised to utilize this approach.

Additionally, note that Spotify's filename recognition systems are case-insensitive, though we recommend lowercase for consistency.

Updates

When making updates for a given GTIN, please take into account the following requirements. Note that updates can **only** be made via your SFTP feed, and that for all asset types, the most recent file provided will override the current record in Spotify's systems.

ASSET TYPE	UPDATES
Book Metadata	Deliver the full metadata. Spotify does not support block updates (partial metadata updates). Spotify does support receipt of metadata without corresponding audio deliveries. However, if you are making updates to the CLM, you ***must*** also redeliver the full audio.
Cover Art	Spotify only supports a single cover image. Any subsequent sends will fully replace the existing cover.
Audio (Audiobook Contents)	If your ONIX contains CLM, all redeliveries of Audiobook Contents ***must*** also include redelivery of the ONIX files. Always re-send all audio files for a given ***audiobook*** - you cannot replace a single audio file. Delivery of a single file may result in the full audiobook contents being replaced with that one file.
Audio (Sample)	Spotify only supports a single sample file - any subsequent sends will fully replace the existing sample.
Supplemental PDF	Any subsequent sends will fully replace the existing PDF.

5. Feed Metadata

Feed metadata identifies the delivery itself. It tells Spotify who sent the file, when it was sent, what type of product it is, and how to associate it with an ***audiobook*** in the catalog.

For an introduction to what feed metadata is and how it fits into a delivery, see Key Concepts. For a complete walkthrough with XML examples, see Getting Started: Delivering Your First Audiobook.

The reference articles below document each feed metadata field in detail, including how Spotify reads the value and fallback behavior when the primary source is missing.

- **Sender**: identifies the organization that sent the file
- **Sent Date Time**: protects against out-of-order processing
- **Product Identifiers**: the ISBN-13 or GTIN-13 that ties the ONIX to the audio and cover art
- **Product Form:** confirms the delivery is an audio product
- **Product Content Type:** confirms the content is an ***audiobook***
- **Notification Type**: tells Spotify what to do with the file (new delivery, update, or ***takedown***)

5.1. Sender

A **<Sender>** ***must*** be provided in the ONIX message header to identify who sent the file. Spotify uses this to verify that the file was delivered to the correct feed.

Send the same sender identification consistently across your deliveries so it can be matched reliably to your provider account. Sender information is not shown to end users.

Spotify Behavior

Spotify reads the **<Sender>** composite from the ONIX message **<Header>** using the following selection path:

1. The **<SenderName>** is read as the primary identifier for the sending organization.

Fallback Logic

PRIORITY	SOURCE	NOTES
1	**<SenderName>**	Use this value when present and non-empty.
2	**<SenderIdentifier>**	Used when **<SenderName>** is absent.

Example

```
<ONIXMessage>
 <Header>
   <Sender>
     <SenderName>Example Audio Publishing</SenderName>
   </Sender>
 </Header>
 <!-- ... -->
</ONIXMessage>
```

Result: The delivery is identified as coming from `Example Audio Publishing`.

5.2. Sent Date Time

A **<SentDateTime>** ***must*** be provided in the ONIX message header for ***ingestion***. Spotify uses **<SentDateTime>** to protect against out-of-order deliveries and to ensure only the most up-to-date ONIX is ingested for a given ***audiobook***. It is expected that each unique delivery will have a unique valid **<SentDateTime>**.

For validation messages and edge cases, see Validation : Dates.

Spotify Behavior

Spotify reads **<SentDateTime>** from the ONIX message **<Header>** using the following selection path:

1. The **<SentDateTime>** value is parsed by attempting each supported date-time format, from most specific to least specific
    a. `YYYYMMDDThhmmss` with timezone
    b. `YYYYMMDDThhmmss` without timezone (UTC is assumed)
    c. `YYYYMMDDThhmm` with timezone
    d. `YYYYMMDDThhmm` without timezone (UTC is assumed)
    e. `YYYYMMDD` (midnight UTC is assumed)

Example

```
<ONIXMessage>
 <Header>
   <SentDateTime>20260717T1300Z</SentDateTime>
 </Header>
 <!-- ... -->
</ONIXMessage>
```

Result: The delivery's send time is recorded. If a newer file has already been received for the same ***audiobook***, this file is skipped as stale; otherwise it is processed normally.

5.3. Product Identifiers

A valid product identifier ***must*** be provided in the ONIX for ***ingestion***. Spotify requires GTIN-13 and/or ISBN-13, but other identifiers will also be stored in Spotify systems.

For validation messages and edge cases, see Validation : Product Identifiers.

Spotify Behavior

Spotify reads every **`<ProductIdentifier>`** composite on the **`<Product>`** using the following selection path:

1. Each **`<ProductIdentifier>`** is checked for a recognized **`<ProductIDType>`** and a non-blank **`<IDValue>`**. [ONIX code list 5]

Accepted Codes

The following codes are accepted. Codes from [ONIX code list 5] not listed here are not supported.

IDENTIFIERS	ACCEPTED CODE VALUES
Product Identifiers	`03, 15, 01`

Example

```
<Product>
 <ProductIdentifier>
   <ProductIDType>15</ProductIDType>
   <IDValue>9781234567890</IDValue>
 </ProductIdentifier>
 <ProductIdentifier>
   <ProductIDType>01</ProductIDType>
   <IDTypeName>MyPublisherSKU</IDTypeName>
   <IDValue>ABC-00123</IDValue>
 </ProductIdentifier>
</Product>
```

Result: ISBN = `9781234567890` Proprietary identifier `MyPublisherSKU` = `ABC-00123`

5.4. Product Form

A **`<ProductForm>`** ***must*** be provided in the ONIX for ***ingestion***. This informs Spotify that the delivery's intended consumption format is that of an ***audiobook***.

Spotify Behavior

Spotify reads **`<ProductForm>`** from **`<DescriptiveDetail>`** using the following selection path:

1. The **`<ProductForm>`** value is checked against the accepted audio forms. [ONIX code list 150]

Accepted Codes

The following codes are accepted. Codes from [ONIX code list 150] not listed here are not supported.

FORMS	ACCEPTED CODE VALUES
Product Form	`AA, AJ, AN, AZ`

Example

```
<DescriptiveDetail>
 <ProductForm>AJ</ProductForm>
</DescriptiveDetail>
```

Result: Accepted as an ***audiobook***. Ingestion proceeds.

5.5. Product Content Type

A **`<PrimaryContentType>`** ***should*** be provided to indicate the nature of the audio content. Spotify uses this to confirm the delivery is an audiobook and not another form of audio such as an audio drama or radio play. If no content type composites are present, the ONIX is assumed to represent an audiobook.

Spotify Behavior

Spotify reads the content type from the **`<DescriptiveDetail>`** block using the following selection path:

1. The **`<PrimaryContentType>`** value is checked against rejected content types. [ONIX code list 81]
2. Each **`<ProductContentType>`** value is checked against rejected content types. [ONIX code list 81]

If neither element is present, the check is passed and ingestion continues. The content type is used only to verify the delivery is an audiobook; it is not stored in the catalog.

Rejected Codes

The following codes from [ONIX code list 81] will cause the delivery to be rejected.

IDENTIFIERS	REJECTED CODE VALUES
Product Content Types	`02`

Example

```
<Product>
 <!-- ... -->
 <DescriptiveDetail>
   <PrimaryContentType>01</PrimaryContentType>
 </DescriptiveDetail>
 <!-- ... -->
</Product>
```

Result: **`<PrimaryContentType>`** `01` (Audiobook) is not a restricted content type. Ingestion proceeds.

5.6. Notification Type

A **`<NotificationType>`** ***must*** be provided, and it must be a valid ONIX code, for the delivery to be ingested. It tells Spotify what action this record is meant to perform, deliver an ***audiobook*** or perform a ***takedown***.

Spotify Behavior

Spotify reads **`<NotificationType>`** directly from **`<Product>`**.

Code `03` (notification confirmed): Any code other than `04` or `05` is processed as a normal record and does not explicitly mark an audiobook as active.

Code `04` (update partial): Spotify does not support partial updates. Redeliveries of ONIX files are expected to be whole representations of the metadata.

Code `05` (delete): Spotify makes the ***audiobook*** inactive. Spotify does not support complete catalog removal via ONIX.

Rejected Codes

The following codes from [ONIX code list 1] will cause the delivery to be rejected. Codes not listed here will be processed.

TYPES	REJECTED TYPE VALUES
Notification Types	`04`

Example of Standard Delivery

```
<Product>
 <NotificationType>03</NotificationType>
 <!-- ... -->
</Product>
```

Result: The record is processed normally.

Example of a Takedown

```
<Product>
 <NotificationType>05</NotificationType>
 <!-- ... -->
</Product>
```

Result: The audiobook is inactive, even if its publishing statuses indicate active.

6. Content Metadata

Content metadata describes the ***audiobook*** itself. It informs the ***audiobook*** detail page experience that end users see in the Spotify application and provides Spotify with signals for discovery and recommendations.

For an introduction to what content metadata is and how it fits into a delivery, see Key Concepts. For a complete walkthrough with XML examples, see Getting Started: Delivering Your First Audiobook
.

The reference articles below document each content metadata field in detail, including how Spotify reads the value, fallback behavior when the primary source is missing, and validation rules.

- **Title**: display name and subtitle handling
- **Description**: consumer-facing summary, text type preference, and language matching
- **Contributors**: authors, narrators, and translators
- **Subjects**: classification schemes and explicit-content flagging
- **Series / Collection**: series title and position
- **Edition**: abridged, unabridged, and other edition types
- **Duration**: ONIX-reported runtime
- **Publisher**: publisher name selection
- **Copyright**: copyright type, year, and owner
- **Chapter-Level Metadata**: chapter names, part groupings, and multi-part audiobooks

6.1. Title

A title ***must*** be provided for the audiobook to be available to ***end users*** on the Spotify application.

If a subtitle is provided, on some surfaces, ***end users*** may see these values displayed together as "`Title: Subtitle`."

It is strongly encouraged that you **do not** include any extraneous non-title information (such as the Series / Collection or Edition) in your title or subtitle, as this may negatively impact the ***end user*** experience and/or result in a truncated title.

For validation messages and edge cases, see Validation : Title.

Spotify Behavior

Spotify reads the audiobook title and subtitle from the **`<DescriptiveDetail>`** block using the following selection path:

1. The first **`<TitleDetail>`** whose **`<TitleType>`** is `01` is selected. All other title types, such as `02`, are ignored. [ONIX code list 15]
2. Within that **`<TitleDetail>`**, the first **`<TitleElement>`** whose **`<TitleElementLevel>`** is `01` (Product level) is selected. Collection-level and other elements are skipped. [ONIX code list 149]
3. The title is then read from **`<TitleText>`**
4. The subtitle is read from **`<Subtitle>`**

Both title and subtitle are normalized according to the Common ONIX Processing Rules : Text Processing.

In addition, trailing articles in the title text are rewritten to leading positions. For example, "Galaxy, The" becomes "The Galaxy". This rule is case-insensitive and applies to the following articles: A, An, The, Das, Der, Die, El, I, Il, L, La, Le, Les, Los, and O.

The title and subtitle's language is determined according to Common ONIX Processing Rules : Language Resolution.

Fallback Logic

PRIORITY	SOURCE	NOTES
1	**`<TitleText>`**	Use this value when it is present and non-empty.

PRIORITY	SOURCE	NOTES
2	**<TitlePrefix>** + **<TitleWithoutPrefix>**	If **<TitleText>** is absent or empty, construct the title as “Prefix Title”. Trailing-article normalization is not applied on this path (the publisher has already separated the prefix).

If **<Subtitle>** is missing or contains no text after whitespace is trimmed, Spotify leaves the subtitle unset.

Example With Subtitle

```
<Product>
 <!-- ... -->
 <DescriptiveDetail>
   <TitleDetail>
     <TitleType>01</TitleType>
     <TitleElement>
       <TitleElementLevel>01</TitleElementLevel>
       <TitleText language="spa">Las vocales piratas</TitleText>
       <Subtitle language="spa">Una lección de respeto y valor</Subtitle>
     </TitleElement>
   </TitleDetail>
 </DescriptiveDetail>
 <!-- ... -->
<Product>
```

Result: Finalized title = `Las vocales piratas: Una lección de respeto y valor`, Title = `Las vocales piratas`, Subtitle = `Una lección de respeto y valor`, Language = `spa`

Example Without Subtitle

```
<Product>
 <!-- ... -->
 <DescriptiveDetail>
   <TitleDetail>
     <TitleType>01</TitleType>
     <TitleElement>
       <TitleElementLevel>01</TitleElementLevel>
       <TitleText language="spa">Las vocales piratas</TitleText>
     </TitleElement>
   </TitleDetail>
 </DescriptiveDetail>
 <!-- ... -->
</Product>
```

Result: Finalized title = `Las vocales piratas`, Title = `Las vocales piratas`, Subtitle = `“”`, Language = `spa`

6.2. Description

A description ***must*** be provided for an ***audiobook*** to be available to ***end users*** on the Spotify application.

The description is the consumer-facing summary that appears on the ***audiobook's*** detail page.
For validation messages and edge cases, see Validation : Description.

Spotify Behavior

Spotify reads the description from the **<CollateralDetail>** block using the following selection path:

1. Each **<TextContent>** composite is evaluated by its **<TextType>** [ONIX code list 153]
2. Within a composite, the first **<Text>** whose language attribute matches the ***audiobook*** language is selected.
    a. If no **<Text>** matches the ***audiobook*** language, a **<Text>** with no language attribute is used if present.

The selected description's text is normalized according to Common ONIX Processing Rules : Text Processing.

The description's language is determined according to Common ONIX Processing Rules : Language Resolution.

Fallback Logic

PRIORITY	SOURCE	NOTES
1	**<TextContent>** with **<TextType>** `01` (Sender-defined text)	First **<Text>** whose language matches the audiobook language; then **<Text>** with no language attribute
2	**<TextContent>** with **<TextType>** `03` (Description)	First **<Text>** whose language matches the audiobook language; then **<Text>** with no language attribute
3	**<TextContent>** with **<TextType>** `02` (Short description / annotation)	First **<Text>** whose language matches the audiobook language; then **<Text>** with no language attribute

Example

```
<DescriptiveDetail>
 <Language>
   <LanguageRole>01</LanguageRole>
   <LanguageCode>spa</LanguageCode>
 </Language>
</DescriptiveDetail>
...
<CollateralDetail>
 <TextContent>
   <TextType>01</TextType>
   <Text language="fre">Texte générique du diffuseur</Text>
 </TextContent>
 <TextContent>
   <TextType>03</TextType>
   <Text language="spa">Descripción del audiolibro</Text>
 </TextContent>
</CollateralDetail>
```

Result: Description = `Descripción del audiolibro`, Language = `spa`.
Note: The `01` composite is skipped because it has no text matching the ***audiobook*** language and no attribute-less fallback text.

6.3. Contributors

At least one author and at least one narrator ***must*** be provided for the audiobook to be available on Spotify. Translators ***should*** be provided when available. Spotify supports multiple contributors per category (e.g. dual narrator productions).

For validation messages and edge cases, see Validation : Contributors.

Spotify Behavior

Spotify reads every contributor from the **<DescriptiveDetail>** block using the following selection path:

1. For each **<Contributor>** the **<ContributorRole>** is inspected for classification of that contributor. [ONIX code list 17]
2. Within each contributor type, contributors are ordered by their overall **<SequenceNumber>**
    a. If repeated <SequenceNumber> element values occur, order cannot be guaranteed.
    b. If no **<SequenceNumber>** is provided for a **<Contributor>** then the contributors are listed in the order they are read
3. The contributor is then read from the **<PersonName>**

All contributor names are subject to normalization per Common ONIX Processing Rules : Text Processing.

Mappings

Each contributor is mapped based on a subset of Roles found in [ONIX code list 17]. Any codes not found below will be ignored.

CONTRIBUTOR TYPE	ACCEPTED ROLE VALUES
Author	`A01, A02, A04, A06, A07, A11, A12, A14, A13, A15, A16, A18, A19, A20, A21, A23, A24, A25, A29, A30, A31, A32, A33, A36, A38, A39, A40`
Narrator	`E01, E03, E04, E05, E06, E07, E08, E09, E99, A43, A44, B22`
Translator	`B06, B08, B10`

Fallback Logic

PRIORITY	SOURCE	NOTES
1	**<PersonName>**	Full display name.
2	**<PersonNameInverted>**	Reformatted from "Last, First" to "First Last". A third comma-separated part is treated as a suffix (e.g., "Tolkien, J. R. R., Jr." becomes "J. R. R. Tolkien Jr.").
3	Assembled name parts	**<TitlesBeforeNames>** + **<NamesBeforeKey>** + **<PrefixToKey>** + **<KeyNames>** + **<NamesAfterKey>** + **<SuffixToKey>** + **<LettersAfterNames>** + **<TitlesAfterNames>**, joined with spaces.
4	**<CorporateName>**	For an organization.
5	**<CorporateNameInverted>**	Reformatted like **<PersonNameInverted>**.
6	**<UnnamedPersons>**	[Code list 19] value is resolved to its heading value and that is used as the contributor's name. (e.g., "Anonymous").

If no valid name can be constructed for a contributor the value of that contributor will default to an empty name. This may cause unintended display behavior for ***end users.***

Example

```
<DescriptiveDetail>
 <Contributor>
   <SequenceNumber>1</SequenceNumber>
   <ContributorRole>A01</ContributorRole>
   <PersonName>Margaret Atwood</PersonName>
 </Contributor>
 <Contributor>
   <SequenceNumber>2</SequenceNumber>
   <ContributorRole>E07</ContributorRole>
   <PersonName>Claire Danes</PersonName>
 </Contributor>
 <Contributor>
   <SequenceNumber>3</SequenceNumber>
```

```
    <ContributorRole>B06</ContributorRole>
    <PersonName>John Jones</PersonName>
  </Contributor>
</DescriptiveDetail>
```

Result: Author = `Margaret Atwood`, Narrator = `Claire Danes`, Description = `Translated by: John Jones`

6.4. Subjects

Subject codes ***should*** be provided to help Spotify classify the ***audiobook***. They are not required for an ***audiobook*** to be available on the Spotify application, but they are critical to help Spotify enrich the ***end user*** experience and surface your ***audiobook*** with the appropriate audiences.

For validation messages and edge cases, see Validation : Subjects.

Spotify Behavior

Spotify reads subject information from the **`<DescriptiveDetail>`** block using the following selection path:

1. For each **`<Subject>`** the **`<SubjectSchemeIdentifier>`** is inspected for classification. [ONIX code list 27]
2. The subject code is read from **`<SubjectCode>`**

Accepted Codes

The following subject schemes are accepted. Schemes from [ONIX code list 27] not listed here are not supported.

SUBJECT	ACCEPTED CODE VALUES
Subject	`10, 93, 29`

Example

```
<DescriptiveDetail>
 <Subject>
   <SubjectSchemeIdentifier>10</SubjectSchemeIdentifier>
   <SubjectCode>FIC028000</SubjectCode>
 </Subject>
 <Subject>
   <SubjectSchemeIdentifier>93</SubjectSchemeIdentifier>
   <SubjectCode>FL</SubjectCode>
 </Subject>
</DescriptiveDetail>
```

Result: BISAC = `[FIC028000]`, Thema = `[FL]`, explicit = `false`

6.5. Series / Collection

If the ***audiobook*** belongs to a series, series information ***should*** be provided so Spotify can group ***audiobooks*** for the ***end user*** experiences and support recommendations. It is ***optional*** and does not block availability.

For validation messages and edge cases, see Validation : Series.

Spotify Behavior

Spotify reads series information from the **<DescriptiveDetail>** block using the following selection path:

1. The first **<Collection>** whose **<CollectionType>** is `10` is selected. [ONIX code list 148]
2. Within that **<Collection>**, the series title is read from the **<TitleElement>** whose **<TitleElementLevel>** is `02`. [ONIX code list 149]
3. If applicable, the part number is read from **<PartNumber>** within the collection's **<TitleElement>**

If both are present, the series title and part number are combined into a single entry "`SeriesTitle #PartNumber`".

The series title's language is determined according to the Common ONIX Processing Rules : Language Resolution.

Fallback Logic

Series Title

PRIORITY	SOURCE	NOTES
1	**<Collection>** with **<CollectionType>** `10` containing a **<TitleElement>** at level `02`	Use this value when present and non-empty.
2	**<TitleElement>** with **<TitleElementLevel>** `02` in the product's own **<TitleDetail>**	Used only when no **<Collection>** is found.

If no series title can be found from either source, no series is stored.

Part Number

PRIORITY	SOURCE	NOTES
1	**<PartNumber>** within the collection's **<TitleElement>**	Use this value when present and non-empty.
2	**<PartNumber>** within the collection's **<TitleDetail>**	
3	**<CollectionSequenceNumber>**	
4	**<PartNumber>** at the product level	Only used when the collection provides a title but no number.

If no part number is found from any source, the series title is accepted without one

Example

```
<DescriptiveDetail>
 <Collection>
   <CollectionType>10</CollectionType>
   <TitleDetail>
     <TitleElement>
       <TitleElementLevel>02</TitleElementLevel>
       <PartNumber>3</PartNumber>
       <TitleText>Harry Potter</TitleText>
     </TitleElement>
   </TitleDetail>
 </Collection>
</DescriptiveDetail>
```

Result: Series = `Harry Potter #3`

6.6. Edition

The edition type ***should*** be provided to indicate how this recording relates to the original work. It is ***optional*** and does not block availability, but it is useful catalog information for ***end users***.

For validation messages and edge cases, see Validation : Edition.

Spotify Behavior

Spotify reads the edition from the **`<DescriptiveDetail>`** block using the following selection path:

1. The first valid **`<EditionType>`** is selected. [ONIX code list 21]

Example

```
<DescriptiveDetail>
 <EditionType>UBR</EditionType>
</DescriptiveDetail>
```

Result: Edition = `Unabridged`

6.7. Duration

A duration ***could*** be provided in the ONIX. The duration is purely informational for Spotify to assist us in ensuring our audio files fully represent your intended delivery. The runtime of the ***audiobook*** displayed to the ***end user*** is calculated using **audio file** data.

For validation messages and edge cases, see Validation : Duration.

Spotify Behavior

Spotify reads duration from the **<DescriptiveDetail>** block using the following selection path:

1. All **<Extent>** composites whose **<ExtentType>** is `09` (duration) [ONIX code list 23] and whose **<ExtentUnit>** is a representation of time [ONIX code list 24] are compared with all other valid **<Extent>** composites standardized to seconds.
2. The duration is read from the **<Extent>** composite with the largest **<ExtentValue>** after standardization.

Mappings

Valid unit values are a subset of the extent unit codes found in [ONIX code list 24]. Any codes not found below will be ignored.

EXTENT UNIT	ACCEPTED UNIT VALUES
Duration	`04, 05, 06, 14, 15, 16`

Example

```
<DescriptiveDetail>
 <Extent>
   <ExtentType>09</ExtentType>
   <ExtentValue>00830</ExtentValue>
   <ExtentUnit>15</ExtentUnit>
 </Extent>
 <Extent>
   <ExtentType>09</ExtentType>
   <ExtentValue>30000</ExtentValue>
   <ExtentUnit>06</ExtentUnit>
 </Extent>
</DescriptiveDetail>
```

Result: ONIX duration = `30600 seconds`

6.8. Publisher

A publisher ***must*** be provided for an ***audiobook*** to be available to ***end users*** on the Spotify application.

The publisher name identifies the organization that published the ***audiobook*** and is used across the catalog.

For validation messages and edge cases, see Validation : Publisher.

Spotify Behavior

Spotify reads the publisher from the **`<PublishingDetail>`** block using the following selection path:

1. The first **`<Publisher>`** whose **`<PublishingRole>`** is `01` is selected. [ONIX code list 45]
2. The publisher name is then read from **`<PublisherName>`**

Example

```
<PublishingDetail>
 <Publisher>
   <PublishingRole>01</PublishingRole>
   <PublisherName>Example Publisher Inc.</PublisherName>
 </Publisher>
</PublishingDetail>
```

Result: Publisher = `Example Publisher Inc.`

6.9. Copyright

A copyright statement ***should*** be provided so Spotify can display copyright information for the audiobook. If provided, copyright statements ***must*** have a valid copyright owner. A copyright statement ***should*** include the corresponding copyright year.

For validation messages and edge cases, see Validation : Copyright.

Spotify Behavior

Spotify reads each copyright statement from the **<PublishingDetail>** block using the following selection path:

1. For each **<CopyrightStatement>** the **<CopyrightType>** is inspected for classification of that copyright type. [ONIX code list 219]
2. Within each **<CopyrightStatement>** an owner is then read from the **<CorporateName>** and is combined with a corresponding **<CopyrightYear>**.

Fallback Logic

PRIORITY	SOURCE	NOTES
1	**<CorporateName>**	Use this value when it is present and non-empty.
2	**<PersonName>**	If **<CorporateName>** is absent **<PersonName>** is used if present and non-empty

If **<CopyrightType>** is missing, Spotify defaults the copyright to a c statement.

Example

```
<PublishingDetail>
 <CopyrightStatement>
   <CopyrightType>C</CopyrightType>
   <CopyrightYear>2026</CopyrightYear>
   <CopyrightOwner>
     <CorporateName>Example Publisher Inc.</CorporateName>
   </CopyrightOwner>
```

```
    <CopyrightOwner>
      <PersonName>John Jones</PersonName>
    </CopyrightOwner>
  </CopyrightStatement>
</PublishingDetail>
```

Result: Finalized Copyright Statement list = `[Example Publisher Inc. 2026]`, Owner Name = `Example Publisher Inc.`, Copyright Year = `2026`, Copyright Type = `C`

6.10. Chapter-Level Metadata

Chapter Level Metadata ***should*** be provided to ensure Chapter titles are displayed as intended. Spotify will ingest user-facing chapter names and, where appropriate, hierarchical “part” groupings and names.

Spotify currently only accepts values via properly formatted ONIX 3.0+ metadata, and is evaluating further alternatives. Contact your account manager if you are only able to deliver CLM via an unsupported method.

Please note that:

- Descriptive chapter names (e.g., 'Chapter 3: A New Beginning') are encouraged over generic labels (e.g., '3') for the best ***end user*** experience.
- CDATA wrapping (e.g., <![CDATA[Chapter 1: The Beginning]]>) is recommended for **`<TitleText>`** and **`<PartNumber>`** values, but not required.
    - This ensures that special characters such as ampersands (&), angle brackets (< >), and quotation marks (“”) are preserved exactly as intended.
- Track sequencing is determined by way of filenaming patterns, and ONIX CLM is used exclusively for determining the user-friendly chapter names of an ***audiobook***.
    - Names are assigned sequentially based on the sequencing of the tracks and the provided ordering of **`<ContentItem>`** composites.
- Spotify currently only supports a 1:1 relationship between Audio Tracks and Chapter Names
- Other forms of CLM, such as chapter descriptions or durations (e.g. **`<AVitem>`**), are accepted but not currently utilized.
- Malformed CLM may result in a failure to ingest the entire **`<Product>`**.

Requirements for Communicating Chapter Names

All values below are required in order to properly ingest publisher provided CLM unless otherwise noted.

ONIX TAG NAME	PARENT	CONTAINS	TYPE	NOTE
`<ContentDetail>`	**`<Product>`**	**`<ContentItem>`**	composite	
`<ContentItem>`	**`<ContentDetail>`**	**`<TitleDetail>`**	composite	One per non-sample audio file.
`<TitleDetail>`	**`<ContentItem>`**	**`<TitleType>`**, **`<TitleElement>`**	composite	

ONIX TAG NAME	PARENT	CONTAINS	TYPE	NOTE
<TitleType>	**<TitleDetail>**	a single code from (15)	element	
<TitleElement>	**<TitleDetail>**	**<TitleElementLevel>**, **<TitleText>**	composite	
<TitleElementLevel>	**<TitleElement>**	A value of 04, representing that this is a chapter (149)	element	Optional. If provided, ***must*** be 04.
<TitleText>	**<TitleElement>**	text (in this case, a chapter name)	element	

Example Title with Chapter Names

```
 <ContentDetail>
  <ContentItem>
    <TitleDetail>
      <TitleType>01</TitleType>
      <TitleElement>
        <TitleElementLevel>04</TitleElementLevel>
        <TitleText><![CDATA[ Chapter 1: The Beginning ]]></TitleText>
      </TitleElement>
    </TitleDetail>
  </ContentItem>
  <ContentItem>
    <TitleDetail>
      <TitleType>01</TitleType>
      <TitleElement>
        <TitleElementLevel>04</TitleElementLevel>
        <TitleText><![CDATA[ Chapter 2: Moving Forward ]]></TitleText>
      </TitleElement>
    </TitleDetail>
  </ContentItem>
 </ContentDetail>
```

Multi-Part Audiobooks

It is optional, but encouraged, to include hierarchical “Part” information when appropriate to a given title, as the Spotify experience does account for this.

When both **<LevelSequenceNumber>** and **<PartNumber>** (P.18.1) are provided, Spotify uses LevelSequenceNumber for ordering and grouping, and PartNumber for the display label.

<LevelSequenceNumber>

<LevelSequenceNumber> within **<ContentItem>** (P.18.1) is used to indicate hierarchical groupings of content items. This is a dot-delimited string of positive integers where each level represents a deeper nesting.

Spotify uses the first component of the **<LevelSequenceNumber>** to determine which "Part" a chapter belongs to within the Spotify listening experience. For example, all chapters with sequences 2.1, 2.2, 2.3 are grouped under Part 2.

Spotify currently only supports a single nest (e.g. Chapters within Parts) at most.

Examples:

- 1.1 — first item within the first top-level item (e.g., Chapter 1 of Part 1)
- 2.4 — fourth item within the second top-level item (e.g., Chapter 4 of Part 2)
- 2.4.1 — first item within the fourth item within the second top-level item (e.g., Verse 1, Chapter 4, Part 2) **not supported**

Numbers ***should*** reflect logical ordering in the ONIX structure, not necessarily the numbering printed in the table of contents. For example, a "Foreword" before Chapter 1 might have **<LevelSequenceNumber>** of 1, making Chapter 1's sequence number 2.

<PartNumber>

<PartNumber> within **<TitleElement>** (P.18.11 / P.6.3) — A free-text label for the part designation. This is the display name shown to end users (e.g., "Part One", "Book Two", "Volume 3"). It sits inside the **<TitleElement>** composite alongside **<TitleText>**.

NOTE: The **<PartNumber>** in this context is the part label within the audiobook itself, not the series/collection part number.

Example Title with Chapter and Part Names

```
<ContentDetail>
  <!-- Part 1, Chapter 1 -->
  <ContentItem>
    <LevelSequenceNumber>1.1</LevelSequenceNumber>
    <TitleDetail>
      <TitleType>01</TitleType>
      <TitleElement>
        <TitleElementLevel>04</TitleElementLevel>
        <TitleText><![CDATA[ Chapter 1: The Beginning ]]></TitleText>
        <PartNumber>Part One</PartNumber>
      </TitleElement>
    </TitleDetail>
  </ContentItem>
  <!-- Part 1, Chapter 2 -->
  <ContentItem>
    <LevelSequenceNumber>1.2</LevelSequenceNumber>
    <TitleDetail>
      <TitleType>01</TitleType>
```

```
      <TitleElement>
        <TitleElementLevel>04</TitleElementLevel>
        <TitleText><![CDATA[ Chapter 2: Moving Forward ]]></TitleText>
        <PartNumber>Part One</PartNumber>
      </TitleElement>
    </TitleDetail>
  </ContentItem>
  <!-- Part 2, Chapter 1 -->
  <ContentItem>
    <LevelSequenceNumber>2.1</LevelSequenceNumber>
    <TitleDetail>
      <TitleType>01</TitleType>
      <TitleElement>
        <TitleElementLevel>04</TitleElementLevel>
        <TitleText><![CDATA[ Chapter 3: A New Part ]]></TitleText>
        <PartNumber>Part Two</PartNumber>
      </TitleElement>
    </TitleDetail>
  </ContentItem>
</ContentDetail>
```

Please reference the ONIX specifications for further detail.

No CLM Provided

If CLM is not provided, names will default to *Track 1* through *Track [N]* according to the sequential listening order indicated in the filenames.

In the event that a provider sends fewer content items than tracks, the provided names will be assigned sequentially, and any remaining audio files will receive the generic track name values described above.

Conversely, if a provider sends more content items than tracks, excess values will be discarded.

Not Supported

- Chapter names via ID3 **`<title>`** tags
- Chapter names via Filenames
- Unique supplemental spreadsheets or manifest file formats

7. Commercial Metadata

Commercial metadata controls the commercial terms of the ***audiobook***. It controls the lifecycle status, the countries in which the ***audiobook*** may be offered, and the suggested pricing.

For an introduction to what commercial metadata is and how it fits into a delivery, see Key Concepts. For a complete walkthrough with XML examples, see Getting Started: Delivering Your First Audiobook.

The reference articles below document each commercial metadata field in detail, including how Spotify reads the value, fallback behavior when the primary source is missing, and validation rules.

- **Audiobook Availability**: lifecycle status, publishing dates, and sale start date resolution
- **Market Availability**: sales rights, and territory resolution
- **Price Acceptance**: which `<Price>` composites are eligible based on type, qualifier, amount, currency, dates, and tax
- **Price Selection**: how the best accepted suggested retail price is chosen for each market

7.1. Audiobook Availability

Audiobook Availability governs how Spotify decides when an ***audiobook*** first becomes active, or if it should be ***inactivated*** via a ***takedown***. It is strongly recommended that you read the key concepts section before proceeding.

Additional fine-grained controls over your availability will be discussed in further detail in the Market Availability documentation.

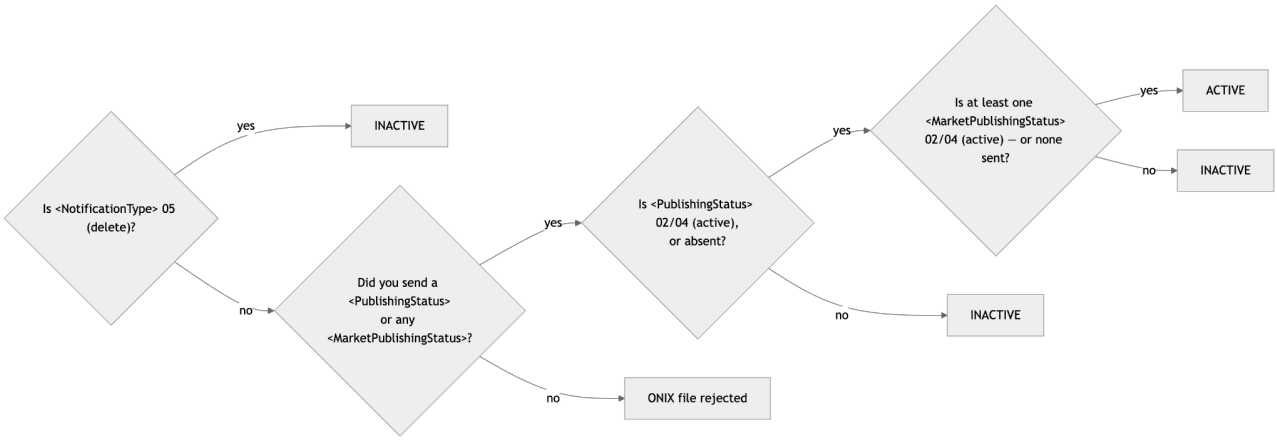
Spotify Behavior

Active Status

Spotify determines whether an ***audiobook*** is active based on the following signals:

1. The notification announces or updates the ***audiobook***. Any **`<NotificationType>`** that Spotify accepts lets the ***audiobook*** be active, except `05` (delete), which makes it inactive. [ONIX code list 1]
2. The product-level status is active or absent. A **`<PublishingStatus>`** with any code other than `02` (forthcoming) or `04` (active) makes the ***audiobook*** inactive. [ONIX code list 64]
3. At least one market-level status is active or none are sent. If any **`<ProductSupply>`** carries a **`<MarketPublishingStatus>`**, one active market (`02`/`04`) is enough to keep the ***audiobook*** active; if all of them are inactive, the ***audiobook*** is ***inactive***. If you send none, the product-level **`<PublishingStatus>`** decides alone.

The ***audiobook*** is active only when all three availability checks align. You ***must*** send at least one of **`<PublishingStatus>`** or **`<MarketPublishingStatus>`**



Sale Start Date

The sale start date is the date from which an ***end user*** may purchase or stream the ***audiobook***. An active ***audiobook*** whose sale start date is still in the future has a ***Countdown Page*** in the Spotify application: ***end users*** can discover it, but can't consume audio until the date passes and the ***audiobook*** reaches ***Full Release***.

Spotify keeps a single sale start date for the whole ***audiobook***, applied to all markets. This date is resolved from three levels of the ONIX record: the product, its markets, and their supplies.

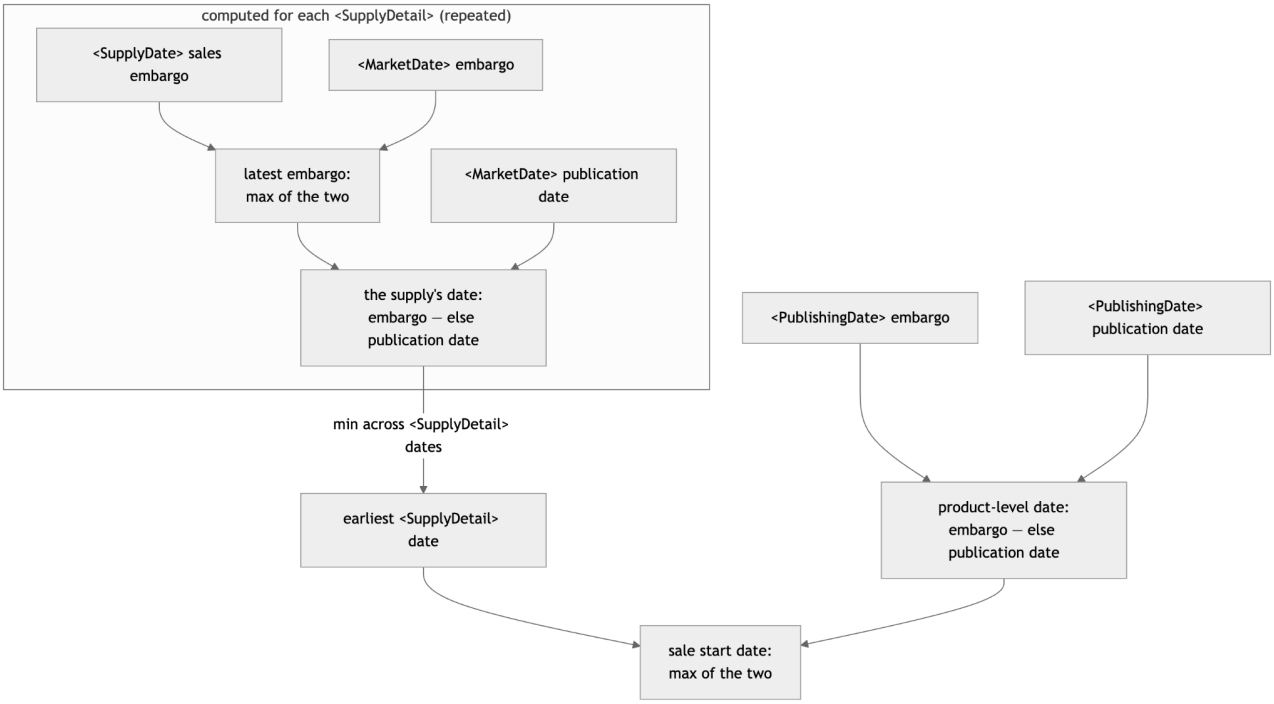
1. Resolve each **`<SupplyDetail>`**'s date. Collect any **`<SupplyDate>`** sales embargo and the **`<MarketDate>`** embargo of its enclosing **`<ProductSupply>`**, and pick the latest. If neither embargo exists, the **`<MarketDate>`** publication date stands in; with no date at all, the **`<SupplyDetail>`** contributes no date.
2. Pick the earliest of those dates, across all **`<SupplyDetail>`** composites.
3. The latest of that date and the resolved **`<PublishingDate>`** is the sale start date.

Only **`<ProductSupply>`** composites that Spotify could actually sell contribute dates: one with no **`<SupplyDetail>`**, for example, will be ignored.

Where the dates sit in the **`<Product>`** record:

```
<Product>
 <!-- ... -->
 <PublishingDetail>
   <PublishingDate><!-- repeatable; role 02 embargo, else role 01 publication
--></PublishingDate>
 </PublishingDetail>
 <ProductSupply> <!-- repeatable -->
   <Market><!-- ... --></Market>
   <MarketPublishingDetail>
     <MarketDate><!-- repeatable; role 02 embargo, else role 01 publication
--></MarketDate>
   </MarketPublishingDetail>
   <SupplyDetail> <!-- repeatable -->
     <SupplyDate><!-- repeatable; role 02 sales embargo only --></SupplyDate>
   </SupplyDetail>
 </ProductSupply>
</Product>
```

How the dates combine:



Takedowns

An ***audiobook*** becomes ***inactive*** if any of the following signals are received:

- **`<NotificationType>`** `05` (delete)
- An ***inactive*** **`<PublishingStatus>`**
- All **`<MarketPublishingStatus>`** values ***inactive***

Spotify doesn't support scheduling a ***takedown*** for a future date. To take an ***audiobook*** down, redeliver the ***audiobook's*** ONIX with an ***inactive*** status.

Example Normal Delivery

```
<Product>
 <NotificationType>03</NotificationType>
 <!-- RecordReference, ProductIdentifier, DescriptiveDetail ... -->
 <PublishingDetail>
   <PublishingStatus>04</PublishingStatus>
   <PublishingDate>
     <PublishingDateRole>01</PublishingDateRole>
     <Date>20250101</Date>
   </PublishingDate>
   <SalesRights>
     <SalesRightsType>01</SalesRightsType>
     <Territory>
       <CountriesIncluded>US GB</CountriesIncluded>
```

```
      </Territory>
    </SalesRights>
  </PublishingDetail>
  <ProductSupply><!-- Market, SupplyDetail, Price ... --></ProductSupply>
</Product>
```

Result: **`<NotificationType>`** `03` marks this as a normal delivery. **`<PublishingStatus>`** `04` means the ***audiobook*** is active, and no **`<MarketPublishingStatus>`** overrides that. The sale start date is 2025-01-01. ***End users*** in the US and GB can purchase or stream the ***audiobook*** from that date.

Example Takedown

```
<Product>
  <NotificationType>03</NotificationType>
  <!-- RecordReference, ProductIdentifier, DescriptiveDetail ... -->
  <PublishingDetail>
    <PublishingStatus>11</PublishingStatus>
    <PublishingDate>
      <PublishingDateRole>01</PublishingDateRole>
      <Date>20250101</Date>
    </PublishingDate>
    <SalesRights>
      <SalesRightsType>01</SalesRightsType>
      <Territory>
        <CountriesIncluded>US GB</CountriesIncluded>
      </Territory>
    </SalesRights>
  </PublishingDetail>
  <ProductSupply><!-- Market, SupplyDetail, Price ... --></ProductSupply>
</Product>
```

Result: **`<NotificationType>`** `03` marks this as a normal delivery, not a delete — but **`<PublishingStatus>`** `11` (withdrawn) is not an active status, so the ***audiobook*** becomes ***inactive*** anyway. The ***takedown*** takes effect the moment Spotify processes this file. A ***takedown*** doesn't require a delete notification; an ***inactive*** status code achieves the same result.

7.1.1. Publishing Status

A **<PublishingStatus>** or **<MarketPublishingStatus>** ***must*** be provided to describe the status of the ***audiobook***. It is recommended to use **<PublishingStatus>** when attempting to represent the status of an ***audiobook*** holistically. A **<MarketPublishingStatus>** can be provided in addition to a **<PublishingStatus>** to describe the status of the ***audiobook*** in a specific market.

Spotify Behavior

Spotify reads **<PublishingStatus>** from **<PublishingDetail>**. Any code other than `02` or `04` will cause a ***takedown*** for the ***audiobook*** in all markets. [ONIX code list 64]

Spotify reads **<MarketPublishingStatus>** from **<MarketPublishingDetail>** within each **<ProductSupply>** composite. Any code other than `02` or `04` will cause a ***takedown*** in the specified market. [ONIX code list 68]

An audiobook takedown will occur if the **<PublishingStatus>** is not `02` or `04`, even if every **<MarketPublishingStatus>** has an active value.

Fallback Logic

If **<PublishingStatus>** is absent, Spotify decides from the market-level **<MarketPublishingStatus>** values only.

Example of an Active Audiobook

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <!-- PublishingDate, SalesRights ... -->
</PublishingDetail>
```

Result: The ***audiobook*** is active.

Example of an Audiobook Inactive in One Market Only

```
<ProductSupply>
 <Market>
   <Territory>
     <CountriesIncluded>US</CountriesIncluded>
   </Territory>
 </Market>
```

```
  <MarketPublishingDetail>
    <MarketPublishingStatus>04</MarketPublishingStatus>
  </MarketPublishingDetail>
  <SupplyDetail><!-- ... --></SupplyDetail>
</ProductSupply>
<ProductSupply>
  <Market>
    <Territory>
      <CountriesIncluded>GB</CountriesIncluded>
    </Territory>
  </Market>
  <MarketPublishingDetail>
    <MarketPublishingStatus>11</MarketPublishingStatus>
  </MarketPublishingDetail>
  <SupplyDetail><!-- ... --></SupplyDetail>
</ProductSupply>
```

Result: The US market is active (`04`); the GB market is inactive (`11` — withdrawn from sale): no new ***end user*** in GB can discover or access the ***audiobook*** once Spotify processes this file. Because at least one market is ***active***, the ***audiobook*** itself stays ***active***, and US is unaffected.

7.1.2. Publishing Date

A PublishingDate composite ***should*** be provided: it contributes to the **sale start date** — the date from which ***end users*** can purchase or stream the ***audiobook***.

Spotify Behavior

Spotify reads **`<PublishingDate>`** composites from **`<PublishingDetail>`**. Role `02` (embargo) is used when present; otherwise role `01` (publication). Any other role is ignored. [ONIX code list 163]

The resolved date combines with any **`<MarketDate>`** and **`<SupplyDate>`** values to determine the whole ***audiobook's*** sale start date. The **`<PublishingDate>`** is the earliest possible date, and Spotify never makes an ***audiobook*** available before it.

See Common ONIX Processing Rules : Date Resolution for supported date formats.

Fallback Logic

1. **`<PublishingDate>`** with role `02` (embargo)
2. **`<PublishingDate>`** with role `01` (publication)
3. Gate availability via **`<MarketDate>`** or **`<SupplyDate>`**

If no valid date for availability is provided, the ***audiobook*** will not be made ***active***.

Example with Publication Date

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <!-- SalesRights ... -->
</PublishingDetail>
```

Result: The sale start date is 2025-01-01.

Example of Embargo Date

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <PublishingDate>
   <PublishingDateRole>02</PublishingDateRole>
   <Date>20250601</Date>
 </PublishingDate>
 <!-- SalesRights ... -->
</PublishingDetail>
```

Result: The sale start date is 2025-06-01 (the embargo date). The publication date of 2025-01-01 is not used.

7.1.3. Market Date

A **<MarketDate>** composite could be provided to delay an audiobook's availability in a specific market. This is useful when a market launch is embargoed beyond the audiobook's publication date.

Spotify Behavior

Spotify reads **<MarketDate>** composites from **<MarketPublishingDetail>** within each **<ProductSupply>** composite. Role `02` (embargo) is used when present. Role `01` (publication date) stands in only when neither the **<MarketDate>** nor the supply's **<SupplyDate>** carries an embargo. Any other role is ignored. [ONIX code list 163]

The resolved date combines with the **<PublishingDate>** and any **<SupplyDate>** values to determine the whole ***audiobook's*** sale start date.

A **<MarketDate>** ***must*** be accompanied by a **<MarketPublishingStatus>** in its **<MarketPublishingDetail>** composite, and a **<Product>** that omits it will be rejected in its entirety.

See Common ONIX Processing Rules : Date Resolution for supported date formats.

Example Market Embargo Delays the Sale Start

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <!-- SalesRights ... -->
</PublishingDetail>
<ProductSupply>
 <Market>
   <Territory>
     <CountriesIncluded>US</CountriesIncluded>
   </Territory>
 </Market>
 <MarketPublishingDetail>
   <MarketPublishingStatus>04</MarketPublishingStatus>
   <MarketDate>
     <MarketDateRole>02</MarketDateRole>
     <Date>20250601</Date>
   </MarketDate>
 </MarketPublishingDetail>
 <SupplyDetail><!-- ... --></SupplyDetail>
</ProductSupply>
```

Result: The sale start date is 2025-06-01. The market embargo pushes the start past the 2025-01-01 publication date and because Spotify keeps one sale start date per ***audiobook***, the start date applies to every market, not just US.

Example Market Date Can't Pull the Start Earlier

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250601</Date>
 </PublishingDate>
 <!-- SalesRights ... -->
</PublishingDetail>
<ProductSupply>
 <Market>
   <Territory>
     <CountriesIncluded>US</CountriesIncluded>
   </Territory>
 </Market>
 <MarketPublishingDetail>
   <MarketPublishingStatus>04</MarketPublishingStatus>
   <MarketDate>
     <MarketDateRole>02</MarketDateRole>
     <Date>20250101</Date>
   </MarketDate>
 </MarketPublishingDetail>
 <SupplyDetail><!-- ... --></SupplyDetail>
</ProductSupply>
```

Result: The sale start date stays 2025-06-01. The **`<PublishingDate>`** is the earliest possible date — the earlier market embargo cannot make the ***audiobook*** available sooner.

7.1.4. Supply Date

A **<SupplyDate>** composite could be provided to delay an audiobook's availability at the supply level with a sales embargo.

Spotify Behavior

Spotify reads **<SupplyDate>** composites from each **<SupplyDetail>**. Only role `02` (sales embargo) is read; every other role is ignored. [ONIX code list 166]

The resolved date combines with the **<PublishingDate>** and any **<MarketDate>** values to determine the whole ***audiobook's*** sale start date.

See Common ONIX Processing Rules : Date Resolution for supported date formats.

Example Supply Embargo Delays the Sale Start

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <!-- SalesRights ... -->
</PublishingDetail>
<ProductSupply>
 <Market>
   <Territory>
     <CountriesIncluded>US</CountriesIncluded>
   </Territory>
 </Market>
 <SupplyDetail>
   <SupplyDate>
     <SupplyDateRole>02</SupplyDateRole>
     <Date>20250601</Date>
   </SupplyDate>
   <!-- Supplier, Price ... -->
 </SupplyDetail>
</ProductSupply>
```

Result: The sale start date is 2025-06-01. The sales embargo pushes the start past the 2025-01-01 publication date, and applies to the whole ***audiobook***.

7.2. Market Availability

Market Availability governs how Spotify interprets publisher intent for which countries an ***audiobook*** may be made available, using ONIX 3.0+ metadata.

Please note that actual on-platform availability is subject to a number of additional factors, such as your contractual agreement with Spotify and the markets in which Spotify offers ***audiobooks***.

Spotify Behavior

Two Required Declarations

Market availability is communicated through two complementary mechanisms in ONIX:

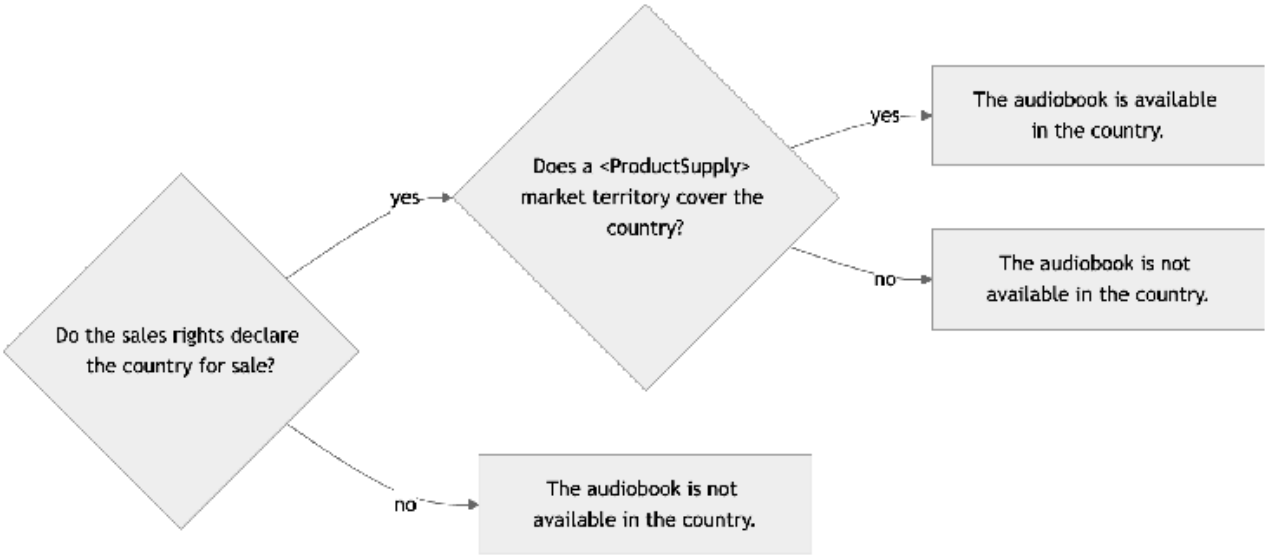
1. **Sales rights** — product-level declarations in `<SalesRights>` of where the ***audiobook*** may or may not be sold. You ***must*** provide `<SalesRights>` or `<ROWSalesRightsType>` to make an ***audiobook*** eligible for a ***full release***.
2. **Product supply** — market-level supply in `<ProductSupply>` that indicates pricing applies in each territory. You ***must*** provide at least one `<ProductSupply>` in order for the ***audiobook*** to receive a ***full release***.

Sales Rights Take Priority

Sales rights define the master set of countries where the ***audiobook*** may be sold. Product supply markets are then intersected with that set — a `<ProductSupply>` can never expand availability beyond what sales rights allow, it can only narrow it.

The interaction works as follows:

1. Sales rights are resolved into a set of available countries (the "sales rights countries"). See Sales Rights for how each entry is read, and Territory for how territories resolve to country lists.
2. For each `<ProductSupply>`, the `<Market>` territory is intersected with the sales rights countries. Any countries in the market territory that are not in the sales rights countries are dropped.
3. A country is only available if it appears in the sales rights countries.



A **`<ProductSupply>`** with no **`<Market>`** covers all sales rights countries — see Market.

An available market can still be ***inactive***: an ***inactive*** **`<MarketPublishingStatus>`** makes the ***audiobook inactive*** in that market even though its countries remain within the sales rights.

Multiple Product Supplies

When multiple **`<ProductSupply>`** composites are present, each is processed independently. All eligible prices from all supplies are collected into a single pool, and the best price is selected per market — see the Price Selection guide. Multiple supplies add to the available options — they do not replace each other.

Example: Product Supply Broader Than Sales Rights

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <CountriesIncluded>US CA GB</CountriesIncluded>
   </Territory>
 </SalesRights>
</PublishingDetail>
<ProductSupply>
 <Market>
   <Territory>
     <RegionsIncluded>WORLD</RegionsIncluded>
   </Territory>
 </Market>
 <SupplyDetail><!-- ... --></SupplyDetail>
```

```
</ProductSupply>
```

The **<ProductSupply>** targets WORLD, but sales rights only authorize US, CA, and GB. **Result:** The ***audiobook*** is available in US, CA, and GB only. All other countries are not available.

Example: Product Supply Narrower Than Sales Rights

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <RegionsIncluded>WORLD</RegionsIncluded>
   </Territory>
 </SalesRights>
</PublishingDetail>
<ProductSupply>
 <Market>
   <Territory>
     <CountriesIncluded>US CA</CountriesIncluded>
   </Territory>
 </Market>
 <SupplyDetail><!-- ... --></SupplyDetail>
</ProductSupply>
```

Sales rights authorize all countries, but the **<ProductSupply>** only targets US and CA. **Result:** US and CA are available with a matching supply. Other countries are authorized by sales rights but have no matching **<ProductSupply>** and will not be available.

Example: Multiple Product Supplies Covering Different Markets

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <CountriesIncluded>US CA GB IE</CountriesIncluded>
   </Territory>
```

```
 </SalesRights>
</PublishingDetail>
<ProductSupply>
 <Market>
   <Territory>
     <CountriesIncluded>US CA</CountriesIncluded>
   </Territory>
 </Market>
 <SupplyDetail><!-- ... --></SupplyDetail>
</ProductSupply>
<ProductSupply>
 <Market>
   <Territory>
     <CountriesIncluded>GB IE</CountriesIncluded>
   </Territory>
 </Market>
 <SupplyDetail><!-- ... --></SupplyDetail>
</ProductSupply>
```

Two **`<ProductSupply>`** composites cover different markets. Both are within the sales rights. **Result:** US, CA, GB, and IE are all available, each matched by their respective **`<ProductSupply>`**.

7.2.1. Sales Rights

Sales rights ***must*** be provided for every country where the ***audiobook*** may be available for ***full release***. This is accomplished by providing detailed **<SalesRights>** composites or allowing rest-of-world sales rights using the **<ROWSalesRightsType>** element instead.

Spotify Behavior

Spotify reads **<SalesRights>** composites from **<PublishingDetail>** to determine which countries your audiobook is authorized to be available. Each **<SalesRights>** entry pairs a **<SalesRightsType>** code with a **<Territory>** that defines the geographic scope. [ONIX code list 46]. Any market in both for-sale and not-for-sale entries will not be allowed.

CODE	RESULTS
00	**<SalesRights>** composite will be ignored. Allows for ROW fallback if **<ROWSalesRightsType>** is provided
01, 02	The territory's countries are available. Spotify does not distinguish between exclusive and non-exclusive rights.
03, 04, 05, 06	The territory's countries are not available.
07, 08	**<SalesRights>** composite will be ignored and does not allow ROW fallback.

```
<Product>
 <!-- ... -->
 <PublishingDetail>
   <SalesRights>
     <Territory><!-- level 3: sales rights countries --></Territory>
   </SalesRights>
 </PublishingDetail>
 <ProductSupply> <!-- repeatable -->
   <Market>
     <Territory><!-- level 2: market territory --></Territory>
   </Market>
   <SupplyDetail> <!-- repeatable -->
     <Price> <!-- repeatable -->
       <Territory><!-- level 1: the price's own territory --></Territory>
     </Price>
   </SupplyDetail>
 </ProductSupply>
</Product>
```

Fallback Logic

For each country, sales rights come from the first source that mentions it:

PRIORITY	SOURCE	NOTES
1	A **<SalesRights>** entry whose territory mentions the country	The entry's type code decides. A country mentioned by any entry — for-sale or not-for-sale — is settled by the **<SalesRights>** composites alone.
2	**<ROWSalesRightsType>** in **<PublishingDetail>**	"ROW" stands for "rest of world": the code applies to every country not mentioned in any **<SalesRights>** territory, using the same code interpretation — `01` or `02` make them available, `03` through `06` make them not available.

If **<ROWSalesRightsType>** is absent, countries not mentioned in any **<SalesRights>** entry have no sales rights declaration and are not available.

Example with Sale in Specific Countries

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <CountriesIncluded>US CA GB</CountriesIncluded>
   </Territory>
 </SalesRights>
</PublishingDetail>
```

Result: The ***audiobook*** is available in the US, CA, and GB.

Example with Sale and Not for Sale

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <CountriesIncluded>US CA GB</CountriesIncluded>
   </Territory>
 </SalesRights>
 <SalesRights>
   <SalesRightsType>04</SalesRightsType>
   <Territory>
     <CountriesIncluded>FR DE</CountriesIncluded>
   </Territory>
 </SalesRights>
</PublishingDetail>
```

Result: The ***audiobook*** is available in the US, CA, and GB. FR and DE are not available. All other countries have no sales rights declared and are not available.

Example with Sale and Not for Sale provided for the same territory

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <CountriesIncluded>US CA GB FR DE</CountriesIncluded>
   </Territory>
 </SalesRights>
 <SalesRights>
   <SalesRightsType>03</SalesRightsType>
   <Territory>
     <CountriesIncluded>FR DE</CountriesIncluded>
   </Territory>
 </SalesRights>
</PublishingDetail>
```

Result: The audiobook is available in US, CA, and GB only. FR and DE are not available because they appear in both **`<SalesRights>`** composites and the not-for-sale takes precedence.

Example with ROWSalesRightsType for Sale

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <ROWSalesRightsType>01</ROWSalesRightsType>
</PublishingDetail>
```

Result: The ***audiobook*** is available in all possible territories.

Example with ROWSalesRightsType with Exclusions

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <CountriesIncluded>US CA</CountriesIncluded>
   </Territory>
 </SalesRights>
 <SalesRights>
   <SalesRightsType>03</SalesRightsType>
   <Territory>
     <CountriesIncluded>FR DE</CountriesIncluded>
   </Territory>
 </SalesRights>
 <ROWSalesRightsType>02</ROWSalesRightsType>
</PublishingDetail>
```

Result: The ***audiobook*** is available in all possible territories except FR and DE.

Example with ROWSalesRightsType Not for Sale

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
```

```
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <CountriesIncluded>US CA GB</CountriesIncluded>
   </Territory>
 </SalesRights>
 <ROWSalesRightsType>03</ROWSalesRightsType>
</PublishingDetail>
```

Result: The ***audiobook*** is available in US, CA, and GB only. The **`<ROWSalesRightsType>`** explicitly excludes everything else.

Example with No Sales Rights

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
</PublishingDetail>
```

Result: No changes to the ***audiobook's*** current availability status. **`<PublishingDetail>`** carries neither a **`<SalesRights>`** entry nor a **`<ROWSalesRightsType>`**, so there are no sales rights to read.

Example with No ROWSalesRightsType and SalesRights provided

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <PublishingDate>
   <PublishingDateRole>01</PublishingDateRole>
   <Date>20250101</Date>
 </PublishingDate>
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <CountriesIncluded>US CA GB</CountriesIncluded>
   </Territory>
 </SalesRights>
</PublishingDetail>
```

Result: The ***audiobook*** is available in the US, CA, and GB. All other countries have no sales rights declaration and are not available.

7.2.2. Territory

Spotify reads the **<Territory>** composite in three places: inside **<SalesRights>**, **<Market>**, and **<Price>**.

- At least one Territory ***must*** be provided inside each Market
- At least one Territory ***must*** be provided inside each SalesRights entry
- A Price ***could*** carry territories to narrow where that price applies.

Spotify Behavior

The same resolution applies in all three contexts — the composite resolves to a flat list of individual ISO 3166-1 alpha-2 country codes:

CountriesIncluded + (all countries if `WORLD` in RegionsIncluded) – **CountriesExcluded**

- **<CountriesIncluded>** is a space-separated list of 2-character country codes. Each recognized code is added to the included set; unrecognized codes are silently ignored. [ONIX code list 91]
- **<RegionsIncluded>** only supports the value `WORLD`, which expands to all ~249 ISO 3166-1 country codes. All other region codes (e.g. `ECZ` for the Eurozone) are silently ignored. [ONIX code list 49]
- **<CountriesExcluded>** is subtracted from the included set. It does **not** imply inclusion of everything else.
- **<RegionsExcluded>** is ignored. Spotify does not support sub-national geographic granularity (e.g. `GB-NIR` for Northern Ireland). To exclude countries, use **<CountriesExcluded>**.

A **<Territory>** with no **<CountriesIncluded>** and no **<RegionsIncluded>** resolves to an empty set.

Example Worldwide

```
<Territory>
 <RegionsIncluded>WORLD</RegionsIncluded>
</Territory>
```

Result: All ~249 countries.

Example Worldwide Except Specific Countries

```
<Territory>
 <RegionsIncluded>WORLD</RegionsIncluded>
 <CountriesExcluded>CN KP</CountriesExcluded>
</Territory>
```

Result: All countries except China (CN) and North Korea (KP). 247 countries.

Example Specific Countries Only

```
<Territory>
 <CountriesIncluded>US CA GB AU NZ</CountriesIncluded>
</Territory>
```

Result: 5 countries — US, CA, GB, AU, NZ.

7.3. Price Acceptance

Price acceptance determines which **<Price>** composites are eligible to set the suggested retail price of your ***audiobook***. A **<Price>** composite that passes every check is classified as accepted and considered during Price Selection.

Each **<Price>** composite sits inside a **<SupplyDetail>** within a **<ProductSupply>**, and specifies an amount, a currency, a type, and optionally a territory and a date range.

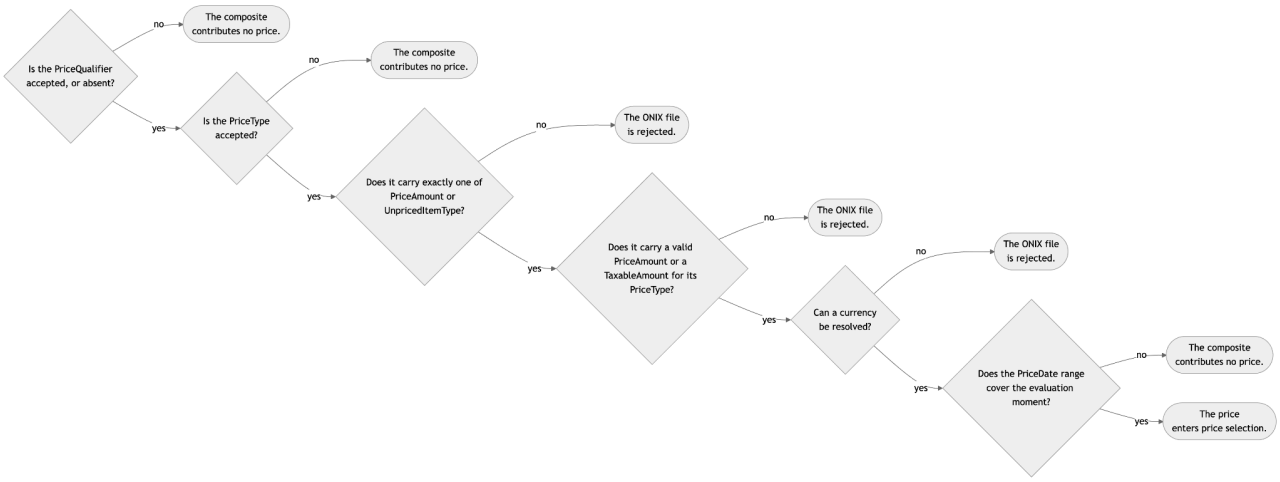
```
<ProductSupply>
 <SupplyDetail> <!-- repeatable -->
   <UnpricedItemType><!-- free/unpriced for the whole supply --></UnpricedItemType>
   <Price> <!-- repeatable -->
     <PriceType><!-- how the price is derived --></PriceType>
     <PriceQualifier><!-- whom the price applies to --></PriceQualifier>
     <PriceAmount><!-- the price value --></PriceAmount>
     <CurrencyCode><!-- ISO 4217; falls back to the header default --></CurrencyCode>
     <UnpricedItemType><!-- free/unpriced for this composite --></UnpricedItemType>
     <Territory><!-- narrows which markets the price applies to --></Territory>
     <PriceDate><!-- repeatable; validity window --></PriceDate>
     <Tax> <!-- repeatable -->
       <TaxableAmount><!-- the price, for tax-inclusive types --></TaxableAmount>
     </Tax>
   </Price>
 </SupplyDetail>
</ProductSupply>
```

Spotify Behavior

Each **<Price>** composite ***must*** carry exactly one price value, either a **<PriceAmount>**, or an **<UnpricedItemType>**. Carrying both, or neither, will cause the ONIX file to be rejected.

A **<Price>** composite is considered for Price Selection only when every check passes:

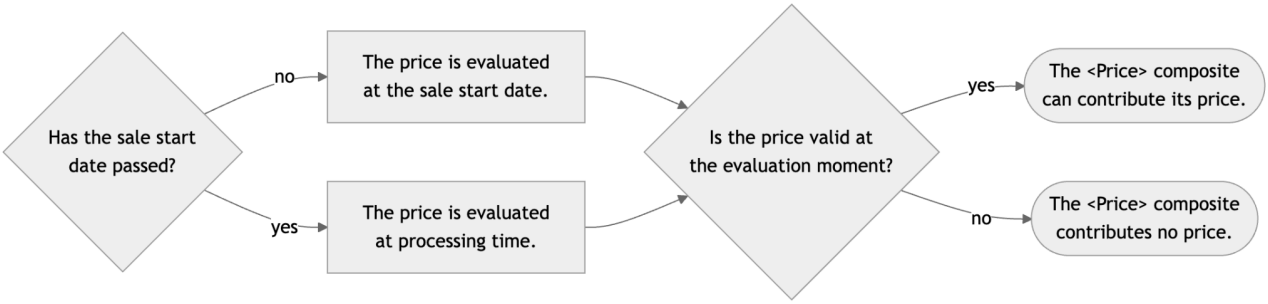
- Its Price Qualifier, when present, marks a price that applies to ***end users***.
- Its Price Type is one of the accepted codes.
  - For the tax-inclusive types (`02`, `42`) the price is the sum of the **<TaxableAmount>** values from the Tax composites — the tax-inclusive **<PriceAmount>** is never used, and the price is not accepted without a **<TaxableAmount>**.
- Its Currency is defined by its own **<CurrencyCode>**, or the **<DefaultCurrencyCode>** from the file's header.
- Its PriceDate is considered acceptable (see below).



Price Date Acceptance

A **`<Price>`** composite ***could*** carry **`<PriceDate>`** composites to limit when its price is valid; each carries a **`<PriceDateRole>`** and a **`<Date>`**. The only accepted role codes for **`<PriceDateRole>`** are `14` and `15`. [ONIX code list 173]

Spotify checks the date range against a single moment: the ***audiobook's*** sale start date or, once that date has passed, the moment it processes the ONIX file. Only unreleased ***audiobooks*** can be given future-dated prices — Spotify does not support scheduling price changes. To change a price later, redeliver the audiobook's ONIX with the new price. See Common ONIX Processing Rules : Date Resolution for supported date formats.



Decision Tree for whether a price is valid at its evaluation moment

If a **`<Price>`** composite includes a **`<Territory>`**, it narrows which markets the price applies to — see Territory. All accepted prices then enter Price Selection, which picks the preferred price per market.

Any other ONIX elements not mentioned above are not supported. This includes, but is not limited to, **`<PriceStatus>`**, **`<PriceIdentifier>`**, **`<PricePer>`**, **`<Discount>`** and **`<DiscountCoded>`**.

Example with Tax-Exclusive and Tax-Inclusive Prices

```
<SupplyDetail>
 <!-- Supplier, ProductAvailability ... -->
 <Price>
   <PriceType>01</PriceType>
   <PriceAmount>14.99</PriceAmount>
   <CurrencyCode>USD</CurrencyCode>
 </Price>
 <Price>
   <PriceType>02</PriceType>
   <PriceAmount>17.09</PriceAmount>
   <Tax>
     <TaxableAmount>14.99</TaxableAmount>
   </Tax>
   <CurrencyCode>USD</CurrencyCode>
 </Price>
</SupplyDetail>
```

Result: Both **`<Price>`** composites pass every check and enter Price Selection. The first is tax-exclusive: the price is 14.99 USD, straight from **`<PriceAmount>`**. The second is tax-inclusive: the price is 14.99 USD from **`<TaxableAmount>`** — its **`<PriceAmount>`** of 17.09 includes tax and is not used.

Example with a PriceDate Range

```
<Price>
 <PriceType>01</PriceType>
 <PriceAmount>9.99</PriceAmount>
 <CurrencyCode>USD</CurrencyCode>
 <PriceDate>
   <PriceDateRole>14</PriceDateRole>
   <Date>20260101</Date>
 </PriceDate>
 <PriceDate>
   <PriceDateRole>15</PriceDateRole>
   <Date>20261231</Date>
 </PriceDate>
</Price>
```

Result: The price is 9.99 USD when an ***audiobook*** is deemed active and the ONIX file was delivered and processed between January 1st 2026 and December 31st 2026 (inclusive). If the ***audiobook*** is forthcoming and its sale start date falls in that window, then the price is 9.99 USD. If the sale start date is outside of the **`<Price>`** composite's window then it will not be considered for Price Selection.

7.3.1. Currency

A currency ***must*** be provided for every priced **<Price>** composite — on the composite itself, or as a file-wide default — and a priced **<Price>** composite with no currency at all will cause the ONIX file to be rejected. A **<Price>** composite carrying an Unpriced Item Type needs no currency.

Spotify Behavior

Spotify reads the currency of a price from the **<CurrencyCode>** on its **<Price>** composite — a three-letter ISO 4217 code. [ONIX code list 96]

Fallback Logic

PRIORITY	SOURCE	NOTES
1	**<CurrencyCode>** on the **<Price>** composite	Used when present — always takes precedence over the default.
2	**<DefaultCurrencyCode>** in the ONIX file's **<Header>**	A file-wide default in the same ISO 4217 form, applying to every **<Price>** composite without a currency of its own.

Example

```
<Header>
 <!-- ... -->
 <DefaultCurrencyCode>USD</DefaultCurrencyCode>
</Header>
<!-- ... -->
<Price>
 <PriceType>01</PriceType>
 <PriceAmount>9.99</PriceAmount>
 <CurrencyCode>GBP</CurrencyCode>
</Price>
<Price>
 <PriceType>01</PriceType>
 <PriceAmount>11.99</PriceAmount>
</Price>
```

Result: The first price is 9.99 GBP — its own **<CurrencyCode>** takes precedence over the default. The second price carries no **<CurrencyCode>**, so its currency comes from **<DefaultCurrencyCode>**: 11.99 USD.

Example with No Currency

```
<Header>
 <!-- no DefaultCurrencyCode -->
</Header>
<!-- ... -->
<Price>
 <PriceType>01</PriceType>
 <PriceAmount>14.99</PriceAmount>
</Price>
<!-- valid price -->
<Price>
 <PriceType>01</PriceType>
 <PriceAmount>9.99</PriceAmount>
 <CurrencyCode>GBP</CurrencyCode>
</Price>
```

Result: The ONIX file will be rejected, even though the second **`<Price>`** composite is valid. The first **`<Price>`** composite has a **`<PriceAmount>`** but no **`<CurrencyCode>`**, and the header carries no **`<DefaultCurrencyCode>`**.

7.3.2. Price Type

A PriceType ***must*** be provided on every priced **<Price>** composite. Only free items declared with an Unpriced Item Type need no PriceType.

Spotify Behavior

Spotify reads **<PriceType>** from each **<Price>** composite to decide whether the price is accepted and how its amount is derived. A **<Price>** composite whose type is missing or not in the accepted codes below contributes no price.

Accepted Codes

A list of accepted price types based on a subset of supported types found in [ONIX code list 58]:

ACCEPTED CODES	BEHAVIOR
01, 41	The price is the **<PriceAmount>**; **<Tax>** composites are ignored.
02, 42	The price is the sum of the **<TaxableAmount>** values from the **<Tax>** composites. For emphasis: these prices will not be accepted unless the **<TaxableAmount>** is present.

For the tax-inclusive types (02, 42) the **<PriceAmount>** includes tax and is not used as the price.

A price whose type is missing or not accepted contributes no price and can never be selected in any market. The type also breaks ties between competing prices.

Price types are considered when determining a Price for the ***audiobook*** during Price Selection.

Example with Tax-Exclusive Price

```
<Price>
 <PriceType>01</PriceType>
 <PriceAmount>14.99</PriceAmount>
 <CurrencyCode>USD</CurrencyCode>
</Price>
```

Result: The price is 14.99 USD, taken directly from **<PriceAmount>**.

Example with Tax-Inclusive Price

```
<Price>
 <PriceType>02</PriceType>
 <PriceAmount>15.74</PriceAmount>
 <Tax>
   <TaxableAmount>14.99</TaxableAmount>
 </Tax>
 <CurrencyCode>USD</CurrencyCode>
</Price>
```

Result: The price is 14.99 USD, from **`<TaxableAmount>`** — not the tax-inclusive **`<PriceAmount>`** of 15.74.

7.3.3. Price Amount

A price amount value ***must*** be provided on every **<Price>** composite without an **<UnpricedItemType>**.

Spotify Behavior

Spotify reads **<PriceAmount>** from each **<Price>** composite that has a **<PriceType>** of `01` or `41` as a non-negative decimal number for the price amount value.

For **<Price>** composites that include a **<Tax>** composite and have a **<PriceType>** of `02` or `42`, the **<PriceAmount>** is not ingested. Instead, at least one **<TaxableAmount>** ***must*** be present for **<Price>** composites with these **<PriceType>** codes. All **<TaxableAmount>** elements will be ingested and then summed to create the price amount value.

Example of Price without Tax

```
<Price>
 <PriceType>01</PriceType>
 <PriceAmount>14.99</PriceAmount>
 <CurrencyCode>USD</CurrencyCode>
</Price>
```

Result: The price is 14.99 USD.

Example of Price with TaxableAmount

```
<Price>
 <PriceType>42</PriceType>
 <PriceAmount>16.48</PriceAmount>
 <Tax>
   <TaxableAmount>10.00</TaxableAmount>
 </Tax>
 <Tax>
   <TaxableAmount>4.99</TaxableAmount>
 </Tax>
 <CurrencyCode>EUR</CurrencyCode>
</Price>
```

Result: The price is 14.99 EUR (10.00 + 4.99).

7.3.4. Price Qualifier

A **<PriceQualifier>** ***could*** be provided. However, a **<Price>** composite with any qualifier that is not pertinent for Spotify (e.g. library channel pricing) is not considered for Price Selection.

Spotify Behavior

Spotify reads **<PriceQualifier>** from each **<Price>** composite. A **<Price>** composite that has no **<PriceQualifier>** is considered by default.

Accepted Codes

A list of accepted qualifiers based on a subset of supported codes found in [ONIX code list 59]:

QUALIFIERS	ACCEPTED CODE VALUES
Price Qualifiers	00, 05

Example

```
<Price>
 <PriceType>01</PriceType>
 <PriceQualifier>05</PriceQualifier>
 <PriceAmount>14.99</PriceAmount>
 <CurrencyCode>USD</CurrencyCode>
</Price>
```

Result: The **<Price>** composite is accepted. The price is 14.99 USD.

7.3.5. Unpriced Item Type

An **`<UnpricedItemType>`** ***should*** be provided as the preferred mechanism to have Spotify offer the ***audiobook*** free of charge.

Alternatively, it is acceptable (but discouraged) to provide a **`<PriceAmount>`** of zero with an associated currency.

Spotify Behavior

`<UnpricedItemType>` of `01` (free of charge) on a **`<SupplyDetail>`** applies to all markets, and sets the price amount value to 0.00.

For more nuanced controls, you may also establish **`<UnpricedItemType>`** of `01` (free of charge) at the **`<Price>`** level.

Any other code causes the price(s) to not be considered for Price Selection. [ONIX code list 57]

Example with Free at the Price Level

```
<Price>
 <PriceType>01</PriceType>
 <UnpricedItemType>01</UnpricedItemType>
</Price>
```

Result: The price is 0.00. No **`<CurrencyCode>`** is required for free items.

Example with Free at the SupplyDetail Level

```
<SupplyDetail>
 <!-- Supplier, ProductAvailability ... -->
 <UnpricedItemType>01</UnpricedItemType>
</SupplyDetail>
```

Result: The whole supply is a free offer, priced 0.00. No **`<Price>`** composites are allowed.

7.4. Price Selection

Price selection determines which of the accepted **<Price>** composites applies in a given market.

Spotify Behavior

Matching Prices to Markets

First, Spotify establishes which Price points may be applicable to any market.

This utilizes up to three different territory lists.

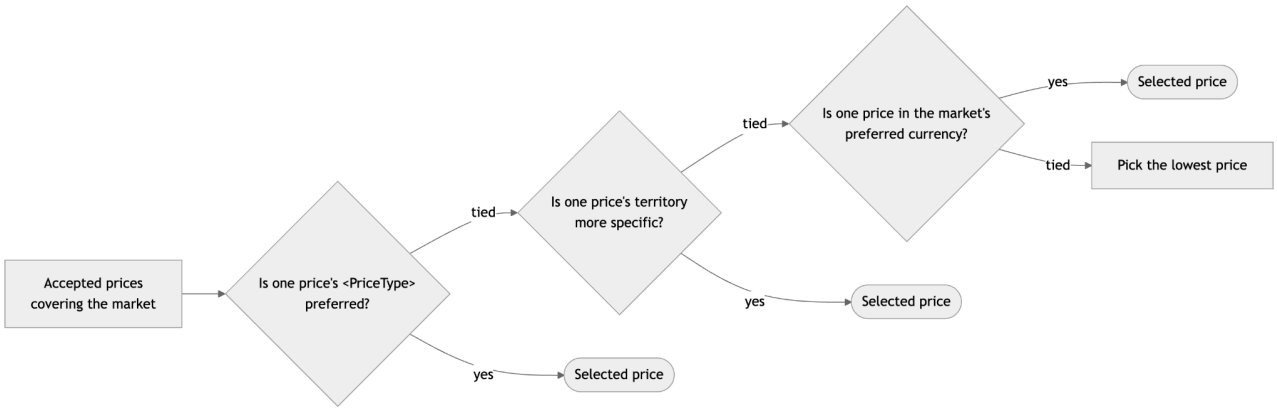
1. **Price**: The **<Territory>** of the **<Price>**
2. **Market:** The **<Territory>** of the **<Market>** in the parent **<ProductSupply>**
3. Sales Rights: The **<Territory>** of the **<SalesRights>** in the **<PublishingDetail>**

A Price is only considered applicable to a market when present in all *provided* territory lists.

In some contexts, providing the above elements is optional. To be considered, a market must be provided in the territory list of *at least one* of the above *or* you must provide **<ROWSalesRightsType>**.

Choosing the Best Price

When multiple accepted prices apply to the same market, Spotify ranks them through four criteria. Each criterion is only used to break a tie from the previous one:



1. PriceType preference

Among the accepted **<PriceType>** codes, Spotify prefers RRP prices over publisher's retail prices, and tax-exclusive over tax-inclusive. [ONIX code list 58]

PREFERENCE	CODE
1	`01`
2	`02`
3	`41`
4	`42`

As a reminder, Spotify only accepts `02` and `42` values if the pre-tax amount can be determined. See Price Acceptance for more details.

2. Territory specificity

If multiple prices are still in consideration for a given market, Spotify's price selection logic will then consider the granularity of the territory data. The more specific a price's territory, the higher its preference:

PREFERENCE	SOURCE	MEANING
1	Price-specific	**`<Price>`** has a **`<Territory>`** covering only a subset of the sales rights countries
2	Price-global	**`<Price>`** has a **`<Territory>`** that covers every sales rights country
3	Market	No **`<Territory>`** on **`<Price>`**; inherits from the **`<Market>`** composite
4	Sales Rights	No **`<Territory>`** on **`<Price>`** and no **`<Market>`** territory

3. Preferred currency

If multiple prices are still in consideration, Spotify's price selection logic will then consider the currency type and whether it is utilized in that market.

A market's preferred currency is the currency Spotify charges ***end users*** in for that market (e.g. USD for US, GBP for GB). A **`<Price>`** in the market's preferred currency is preferred over one in a different currency.

4. Lowest price

Finally, if multiple prices are still in consideration for a given market, the lowest wins — comparing the amount Spotify derives from each accepted price, whatever its **`<PriceType>`**. If needed, amounts are converted to the market's preferred currency for comparison. Free items are 0.00 and therefore always the

lowest.

Example with Prices from Multiple Supplies Competing for One Market

```
<Product>
 <!-- NotificationType, RecordReference, ProductIdentifier, DescriptiveDetail,
      PublishingDetail ... -->
 <ProductSupply>
   <Market>
     <Territory>
       <RegionsIncluded>WORLD</RegionsIncluded>
     </Territory>
   </Market>
   <SupplyDetail>
     <!-- Supplier, ProductAvailability ... -->
     <Price>
       <PriceType>01</PriceType>
       <PriceAmount>12.99</PriceAmount>
       <CurrencyCode>USD</CurrencyCode>
     </Price>
   </SupplyDetail>
 </ProductSupply>
 <ProductSupply>
   <Market>
     <Territory>
       <CountriesIncluded>US</CountriesIncluded>
     </Territory>
   </Market>
   <SupplyDetail>
     <!-- Supplier, ProductAvailability ... -->
     <Price>
       <PriceType>01</PriceType>
       <PriceAmount>14.99</PriceAmount>
       <CurrencyCode>USD</CurrencyCode>
       <Territory>
         <CountriesIncluded>US</CountriesIncluded>
       </Territory>
     </Price>
   </SupplyDetail>
 </ProductSupply>
</Product>
```

Result: For the US market, both prices (12.99 USD and 14.99 USD) compete. Accepted prices from every **`<ProductSupply>`** are collected into a single pool. Both are **`<PriceType>`** `01`, but the 14.99 price carries its own **`<Territory>`** targeting US. It is more territory-specific and is selected. The price for US is 14.99 USD. In every other sales rights country, only the WORLD supply applies at 12.99 USD.

Example with a WORLD Price Resolving to Sales Rights Countries Only

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <!-- PublishingDate ... -->
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <CountriesIncluded>US CA</CountriesIncluded>
   </Territory>
 </SalesRights>
</PublishingDetail>
<ProductSupply>
 <Market>
   <Territory>
     <CountriesIncluded>US CA</CountriesIncluded>
   </Territory>
 </Market>
 <SupplyDetail>
   <!-- Supplier, ProductAvailability ... -->
   <Price>
     <PriceType>01</PriceType>
     <PriceAmount>14.99</PriceAmount>
     <CurrencyCode>USD</CurrencyCode>
     <Territory>
       <RegionsIncluded>WORLD</RegionsIncluded>
     </Territory>
   </Price>
 </SupplyDetail>
</ProductSupply>
```

Result: The **`<Price>`**'s **`<Territory>`** of WORLD resolves to US and CA only — prices are only resolved where you have sales rights.

Example with Prices From Different ProductSupply Composites

```
<ProductSupply>
 <Market>
   <Territory>
     <CountriesIncluded>US CA</CountriesIncluded>
   </Territory>
 </Market>
 <SupplyDetail>
   <!-- Supplier, ProductAvailability ... -->
   <Price>
     <PriceType>01</PriceType>
     <PriceAmount>14.99</PriceAmount>
     <CurrencyCode>USD</CurrencyCode>
   </Price>
 </SupplyDetail>
</ProductSupply>
<ProductSupply>
 <Market>
   <Territory>
```

```
      <CountriesIncluded>GB IE</CountriesIncluded>
    </Territory>
  </Market>
  <SupplyDetail>
    <!-- Supplier, ProductAvailability ... -->
    <Price>
      <PriceType>01</PriceType>
      <PriceAmount>11.99</PriceAmount>
      <CurrencyCode>GBP</CurrencyCode>
    </Price>
  </SupplyDetail>
</ProductSupply>
```

Result: For US, 14.99 USD is selected (from the first **`<ProductSupply>`**); for GB, 11.99 GBP (from the second). Each market is matched to the **`<ProductSupply>`** whose territory covers it.

Example with PriceType Winning Over Territory Specificity

```
<!-- Price A: PriceType 01, targets WORLD -->
<Price>
 <PriceType>01</PriceType>
 <PriceAmount>14.99</PriceAmount>
 <CurrencyCode>USD</CurrencyCode>
 <Territory>
   <RegionsIncluded>WORLD</RegionsIncluded>
 </Territory>
</Price>
<!-- Price B: PriceType 41, targets US specifically -->
<Price>
 <PriceType>41</PriceType>
 <PriceAmount>12.99</PriceAmount>
 <CurrencyCode>USD</CurrencyCode>
 <Territory>
   <CountriesIncluded>US</CountriesIncluded>
 </Territory>
</Price>
```

Result: For the US market, Price A (14.99 USD) is selected. PriceType `01` outranks `41` even though Price B is more territory-specific and cheaper.

Example with Territory Specificity Breaking a PriceType Tie

```
<!-- Price A: PriceType 01, targets WORLD -->
<Price>
 <PriceType>01</PriceType>
 <PriceAmount>14.99</PriceAmount>
```

```
 <CurrencyCode>USD</CurrencyCode>
 <Territory>
   <RegionsIncluded>WORLD</RegionsIncluded>
 </Territory>
</Price>
<!-- Price B: PriceType 01, targets US specifically -->
<Price>
 <PriceType>01</PriceType>
 <PriceAmount>12.99</PriceAmount>
 <CurrencyCode>USD</CurrencyCode>
 <Territory>
   <CountriesIncluded>US</CountriesIncluded>
 </Territory>
</Price>
```

Result: For the US market, Price B (12.99 USD) is selected. Both are PriceType `01`, but Price B is more territory-specific (targets US directly vs WORLD).

Example with Authorized Market With No Price

```
<PublishingDetail>
 <PublishingStatus>04</PublishingStatus>
 <!-- PublishingDate ... -->
 <SalesRights>
   <SalesRightsType>01</SalesRightsType>
   <Territory>
     <CountriesIncluded>US GB</CountriesIncluded>
   </Territory>
 </SalesRights>
</PublishingDetail>
<ProductSupply>
 <Market>
   <Territory>
     <CountriesIncluded>US GB</CountriesIncluded>
   </Territory>
 </Market>
 <SupplyDetail>
   <!-- Supplier, ProductAvailability ... -->
   <Price>
     <PriceType>01</PriceType>
     <PriceAmount>14.99</PriceAmount>
     <CurrencyCode>USD</CurrencyCode>
     <Territory>
       <CountriesIncluded>US</CountriesIncluded>
     </Territory>
   </Price>
 </SupplyDetail>
</ProductSupply>
```

Result: For US, 14.99 USD is selected. GB is authorized by sales rights and covered by the **`<Market>`**, but the only price's **`<Territory>`** resolves to US alone — no price covers GB, so ***end users*** in GB cannot purchase or stream the audiobook. The ONIX file is accepted.

Appendix

The appendix contains supplemental reference material that supports the main specification. These articles are not required reading for delivering an ***audiobook***, but provide detailed technical information for troubleshooting and integration.

- **Validation**: every validation message Spotify may produce during ***ingestion***, organized by concept. Use this to diagnose warnings and errors.
- **Common ONIX Processing Rules**: text normalization, language resolution, and shared processing behavior that applies across multiple metadata fields.
- **Example Audiobook**: a complete, valid ONIX file for a single ***audiobook*** delivery.

A.1. Validation

This section catalogs the alerts that may surface in validation. For the correct delivery format and fallback logic, refer to the matching reference article elsewhere in this specification.

How to Use This Section

1. Locate the concept matching the field in question (e.g., Title, Description, Contributors).
2. Review the alert table for the message received.
3. Read the **Condition** column to understand what triggered the alert.
4. Read the **Impact** column to understand the downstream effect.

As a reminder, when reviewing impact statements, any alert preventing availability of a ***Countdown Page*** will also prevent a ***Full Release***.

Shared Alerts and Edge Cases

These alert types are shared across multiple text fields (title, subtitle, description, contributors, series name, etc.) and are documented here once rather than repeated in each concept section.

Language Resolution

Alerts

ALERT CONDITION	MESSAGE	IMPACT
A text element (e.g., **`<TitleText>`**, **`<Subtitle>`**, **`<Text>`**) has no `language` attribute	`{Field}` element has no language attribute. Falling back to audiobook language `{languages}`.	Processing continues. The product-level language from **`<Language>`** composites is used instead. If no product-level language exists, falls back to header **`<DefaultLanguageOfText>`**, then `'und'` (undetermined).

Resolution: Add a `language` attribute to the element (e.g., **`<TitleText language="eng">`**). This is recommended for multi-language catalogs and eliminates ambiguity.

See Common ONIX Processing Rules : Language Resolution for the full priority chain.

Dates

Date composites appear across multiple levels of the ONIX record (Publishing Date, Market Date, Supply Date, etc). The alerts for dates are documented here once rather than repeated in each concept section.

Alerts

ALERT CONDITION	MESSAGE	IMPACT
The **<Date>** value is empty	**<Date>** element value is empty	The ***audiobook*** fails ***ingestion***.
The **<Date>** cannot be parsed for its declared format	Could not parse date `{value}` with format `{format}`	The ***audiobook*** fails ***ingestion***.
The declared **<DateFormat>** is not a recognized ONIX code	date_format must be a valid ONIX code when present	The ***audiobook*** fails ***ingestion***.
The **<DateFormat>** is a valid code but not one Spotify supports (only `00`/`13`/`14`)	Unsupported date format: `{format}`	The ***audiobook*** fails ***ingestion***.

Edge Cases

SCENARIO	BEHAVIOR
Only a role `02` (sales embargo) date is delivered	Not used as a publication date; treated as no publication date.
<DateFormat> omitted	`YYYYMMDD` is assumed.

See Common ONIX Processing Rules : Date Resolution for the full priority chain.

Alerts and Edge Cases by Reference

The alerts below are specific to individual metadata concepts. Each subsection corresponds to a reference article in this specification.

Product Identifiers

Concept article: Product Identifiers

Alerts

ALERT CONDITION	MESSAGE	IMPACT
A **`<ProductIdentifier>`** has a missing or blank **`<IDValue>`**	Product identifier value is missing or blank - dropping identifier	That identifier is dropped.
A **`<ProductIdentifier>`** has an unrecognized **`<ProductIDType>`** code	Unrecognized **`<ProductIDType>`** code for identifier with value `{value}` - skipping	That identifier is dropped.
An ISBN-13 or GTIN-13 is not exactly 13 digits (or contains punctuation/spaces)	`{ISBN-13|GTIN-13}` should contain exactly 13 digits with no punctuation - found: `{value}` - dropping identifier	That identifier is dropped.
A proprietary identifier (`01`) is missing **`<IDTypeName>`**	Proprietary product ID scheme missing **`<IDTypeName>`** - dropping identifier	That identifier is dropped.
Two proprietary identifiers share the same name and value	Duplicate proprietary identifier with type name `{name}` and value `{value}` - dropping duplicate	The duplicate is dropped.
More than one identifier of the same type is present	Multiple identifiers of type `{type}` found - typically only one identifier per type is expected	All are kept.
An ISBN-13 is present but no matching GTIN-13	Product has ISBN-13 but missing corresponding GTIN-13 identifier - ONIX specification requires ISBN-13 to also be sent as GTIN-13 (code `03`) for book trade transactions	Spotify derives the ISBN from the ISBN-13 alone, so a matching GTIN-13 is optional.
No valid product identifier remains after processing (none delivered, or all delivered identifiers were dropped)	At least one product identifier is required by ONIX specification	The ***audiobook*** fails ***ingestion***.

Edge Cases

SCENARIO	BEHAVIOR
Multiple ISBN-13 or GTIN-13 identifiers	All are kept; the ISBN is taken from the first ISBN-13.
A recognized non-ISBN/GTIN type	Stored under its ONIX type name.

SCENARIO	BEHAVIOR
Only malformed identifiers delivered	All are dropped, which leaves none valid, so the ***audiobook*** fails ***ingestion***.

Title

Concept article: Title

Alerts

ALERT CONDITION	MESSAGE	IMPACT
No **<TitleDetail>** with **<TitleType>** 01 (Distinctive title) found in the product	No **<TitleDetail>** with <TitleType> 01 (Distinctive title) found. The title will be empty.	Spotify cannot locate a title source. The title will be empty. The ***audiobook*** does not receive a ***countdown page***.
More than one **<TitleDetail>** with **<TitleType>** 01 (Distinctive title); only the first is used	Found {n} TitleDetail composites with TitleType 01 (Distinctive title). Only the first is used for title extraction; {n-1} additional TitleDetail(s) are ignored.	Any title data in the additional distinctive **<TitleDetail>** composites are ignored, including subtitle.
<TitleStatement> is present in the TitleDetail (fires regardless of whether TitleText exists)	TitleStatement is present but not used for title extraction. Individual TitleElement data is used instead.	**<TitleStatement>** is valid ONIX but Spotify does not read it; including it alongside **<TitleText>** is harmless but it cannot serve as a title source. Processing continues normally using **<TitleElement>** data.
The distinctive **<TitleDetail>** has no **<TitleElement>** at all	TitleDetail with TitleType 01 has no TitleElement. At least one TitleElement is expected per ONIX 3.0 spec.	No title source in this detail. The title will be empty. The ***audiobook*** does not receive a ***countdown page***.
No product-level (**<TitleElementLevel>** 01) **<TitleElement>** carries **<TitleText>** or **<TitleWithoutPrefix>**	No product-level (TitleElementLevel 01) TitleElement with TitleText or TitleWithoutPrefix found. Title will be empty.	Spotify only reads the title from a product-level element with title content. If none exists, the title is empty and the ***audiobook*** does not receive a ***countdown page***.

ALERT CONDITION	MESSAGE	IMPACT
Multiple **<TitleElement>** blocks at product level with title content; only the first is used	Found {n} product-level TitleElement composites with title text. Only the first is used for title extraction; {n-1} additional TitleElement(s) are ignored.	Data in subsequent product-level elements (including their subtitles) are ignored. If the intended title or subtitle is in a later element, it will not be ***ingested***. Ensure the intended title is in the first **<TitleElement>**.
The **<TitleElement>** used for the title has no language attribute	**<TitleElement>** has no language attribute. Falling back to audiobook language {languages}.	The title is still used; its language is inherited from the resolved product-level language rather than a title-specific tag.

Edge Cases

SCENARIO	BEHAVIOR
<TitleText> present but containing only ASCII whitespace (spaces, tabs, newlines)	Treated as having no value (hasValue() trims ASCII whitespace). The **<TitlePrefix>** + **<TitleWithoutPrefix>** fallback is attempted; if that also yields nothing, the title is empty and the ***audiobook*** will not get a ***countdown page***.
Title resolves to an empty string after all fallbacks	The ***audiobook*** will not get a ***countdown page***.
<Subtitle> element is absent	Normal for an optional field. No subtitle is set.
<Subtitle> is present but blank (whitespace only)	Treated as absent. No subtitle is set.
Subtitle is not in the first **<TitleElement>**	Not found. Spotify reads the subtitle from the first **<TitleElement>** no subtitle is set.

Description

Concept article: Description

Alerts

ALERT CONDITION	MESSAGE	IMPACT
A **<TextContent>** composite has no **<TextType>** element	Dropping TextContent composite at index {n}: TextType element is missing. TextType is mandatory per ONIX 3.0 spec (P.14.1).	That composite is ignored. Other composites are still evaluated.
A **<TextContent>** **<TextType>** value is not in [ONIX code list 153]	Dropping TextContent composite at index {n}: TextType value is not a recognized code from [ONIX code list 153].	That composite is ignored. Other composites are still evaluated.
One or more composites use a recognized but unsupported **<TextType>** (e.g., 04 Table of contents)	Dropped {n} TextContent composite(s) with unsupported TextType. Spotify only uses TextType 01 (Sender defined text), 03 (Description), and 02 (Short description annotation). Dropped: {details}.	Those composites are not used for the description. Supported composites are still evaluated.
A **<Text>** value is null, blank, or Spotify's parser returns serialized XML	Skipping invalid Text value: isNull={bool}, isBlank={bool}, serializedXml={bool}.	That **<Text>** element is skipped. Other **<Text>** elements are still evaluated.
No **<Text>** matches the audiobook language, but a **<Text>** with no language attribute exists	No exact language match found for languages {languages}. Using Text element with no language attribute as fallback.	The attribute-less text is used; its language is inherited from the audiobook. Processing continues.
Description text required more than one HTML entity decode pass	Description text required {n} HTML entity decode passes (multiply-encoded entities detected).	The text is fully decoded. Informational only.

Edge Cases

SCENARIO	BEHAVIOR
Text contains XHTML formatting tags (**<p>**, ****, etc.)	Preserved as delivered.
Text is wrapped in CDATA	Unwrapped; the content is preserved.
Entities that have been encoded multiple times (e.g., &)	Decoded iteratively until stable, up to 5 passes. A warning fires if more than one productive pass was needed.

SCENARIO	BEHAVIOR
Description exceeds any length	Accepted without modification. No max-length enforcement.

Contributors

Concept article: Contributors

Alerts

ALERT CONDITION	MESSAGE	IMPACT
A **<Contributor>** has a **<ContributorRole>** that is not categorized into the accepted roles.	Contributor with name `{name}` has no accepted roles	That contributor is dropped from all lists.
No **<Contributor>** has an author role, or every author's name resolves empty	No contributors with author roles present.	No author is set. The ***audiobook*** will not receive a ***countdown page***.
No **<Contributor>** has a narrator role, or every narrator's name resolves empty	No contributors with narrator roles present.	No narrator is set. The ***audiobook*** will not be available for ***full release***.

Edge Cases

SCENARIO	BEHAVIOR
A contributor has multiple roles spanning categories	Appears in each matching list (e.g., both author and narrator).
Multiple authors or narrators	All kept, ordered by **<SequenceNumber>**; those without a sequence number are ordered last.
Corporate contributor (no person-name elements)	Name resolved from **<CorporateName>** / **<CorporateNameInverted>**.
Duplicate author or narrator names	Deduplicated before the names are written to the catalog.

Subjects

Concept article: Subjects

ALERT CONDITION	MESSAGE	IMPACT
A **<Subject>** has no **<SubjectSchemeIdentifier>**	**<Subject>** missing **<SubjectSchemeIdentifier>**	That subject is dropped.
A **<Subject>** has a missing or blank **<SubjectCode>**	**<Subject>** missing **<SubjectCode>**	That subject is dropped.
One or more codes are in an unsupported scheme (e.g., BIC, keywords)	Dropped `{n}` unsupported `{scheme}` subject code(s): `{codes}`	Those subjects are dropped; BISAC/Thema/CLIL codes are unaffected.

Edge Cases

SCENARIO	BEHAVIOR
Duplicate codes within a scheme	Deduplicated.
Lowercase or mixed-case codes	Uppercased.
No subject codes delivered	No codes stored

Series / Collection

Concept article: Series / Collection

Alerts

No alerts at this time.

Edge Cases

SCENARIO	BEHAVIOR
Multiple **<Collection>** composites	The first with a usable collection is chosen.
Series title present but no part number	The series information will pass ***ingestion*** but with no part number.

Edition

Concept article: Edition

Alerts

ALERT CONDITION	MESSAGE	IMPACT
More than one **<EditionType>** is present	More than one **<EditionType>** tag found. Using the first valid tag.	The first valid one is used for ***ingestion***.
An **<EditionType>** code is not in [ONIX code list 21]	Unknown edition type code not found in ONIX Code List 21	No Edition is set
A valid but less common edition type is used	Newly supported edition type encountered: code=`{code}`, description=`{description}`	The type is accepted and stored.
Both **<NoEdition/>** and one or more **<EditionType>** are present	NoEdition tag and EditionTypes tag(s) found	The ***audiobook*** fails ***ingestion***.

Edge Cases

SCENARIO	BEHAVIOR
No **<EditionType>** and no **<NoEdition/>**	Edition left empty; no message.
<NoEdition/> alone (no **<EditionType>**)	Edition left empty; no message.

Duration

Concept article: Duration

Alerts

CONDITION	MESSAGE	IMPACT
A duration **<Extent>** value/unit cannot be parsed	Skipping unparseable extent: `{reason}`	That extent is dropped. If another parses, it is used; otherwise no ONIX duration is stored.

Edge Cases

SCENARIO	BEHAVIOR
Multiple duration extents	The largest successfully parsed value is used.
Extents of a type other than `09` (Duration)	That duration is dropped.
A duration extent's **`<ExtentUnit>`** is outside the supported set (`04`, `05`, `06`, `14`, `15`, `16`)	Treated as unparseable; the extent is dropped.

Publisher

Concept article: Publisher

Alerts

No alerts at this time.

Edge Cases

SCENARIO	BEHAVIOR
Multiple **`<Publisher>`** composites have role `01`	The first valid one is used for ***ingestion***.
A **`<Publisher>`** is present but its **`<PublishingRole>`** is not `01` (e.g., `02` Co-publisher)	Not selected. If no role-`01` publisher exists, the publisher is empty and the ***audiobook*** will not receive a ***countdown page***.
`<PublisherName>` is present but the composite has no **`<PublishingRole>`**	That publisher is dropped.

Copyright

Concept article: Copyright

Alerts

ALERT CONDITION	MESSAGE	IMPACT
<CopyrightType> is absent	**<CopyrightType>** element is absent. Per ONIX 3.0 spec (P.20.6a), the default when omitted is copyright (C). Defaulting to type C.	Type defaults to C.
<CopyrightType> is P (phonogram right)	**<CopyrightType>** P (phonogram right) is a valid ONIX code (code list 219) but is not supported by Spotify. Dropping this **<CopyrightStatement>**.	The statement is dropped.
<CopyrightType> is D (design right)	**<CopyrightType>** D (design right) is a valid ONIX code (code list 219) but is not supported by Spotify. Dropping this **<CopyrightStatement>**.	The statement is dropped.
<CopyrightType> has an unrecognized code	Unrecognized **<CopyrightType>** code {code}. Expected C (copyright), P (phonogram), or D (design right) per ONIX code list 219. Defaulting to UNKNOWN.	Kept with unknown type.
<CopyrightType> present but not a recognized code	**<CopyrightType>** element is present but the value is not a recognized code from ONIX code list 219. Defaulting to UNKNOWN.	Kept with unknown type.
Statement has neither **<CopyrightYear>** nor **<CopyrightOwner>**	**<CopyrightStatement>** has neither **<CopyrightYear>** nor **<CopyrightOwner>**. At least one is required per ONIX 3.0 spec (P.20).	The statement is dropped.
Statement has a year but no **<CopyrightOwner>**	Dropping **<CopyrightStatement>** that has **<CopyrightYear>** but no **<CopyrightOwner>** composite. Spotify requires an owner name to display copyright information.	The statement is dropped.

ALERT CONDITION	MESSAGE	IMPACT
Owner has only a **<CopyrightOwnerIdentifier>** (no name)	Dropping **<CopyrightStatement>** where **<CopyrightOwner>** contains only a **<CopyrightOwnerIdentifier>** with no **<CorporateName>** or **<PersonName>**. Spotify requires a display name and does not use **<CopyrightOwnerIdentifier>**.	The statement is dropped.
Owner has no name and no identifier	Dropping **<CopyrightStatement>** where **<CopyrightOwner>** has no name (**<CorporateName>** or **<PersonName>**) and no **<CopyrightOwnerIdentifier>**. Per ONIX 3.0 spec, each **<CopyrightOwner>** must carry a name, identifier, or both.	The statement is dropped.
Owner has both **<CorporateName>** and **<PersonName>**	**<CopyrightOwner>** at index {n} has both **<CorporateName>** and **<PersonName>**. Per ONIX 3.0 spec (P.20.11, P.20.12), each **<CopyrightOwner>** should carry a single name (personal or corporate). Using **<CorporateName>**.	**<CorporateName>** is used.
Owner carries **<CopyrightOwnerIdentifier>**	Dropping {n} **<CopyrightOwnerIdentifier>** on **<CopyrightOwner>** at index {i}. Spotify does not use CopyrightOwnerIdentifier — only **<CorporateName>** and **<PersonName>** are used for display.	Identifiers ignored; statement kept if it has a name.

Edge Cases

SCENARIO	BEHAVIOR
Multiple **<CopyrightStatement>** composites	Each is processed independently; all that resolve to an owner are kept.
No **<CopyrightStatement>** at all	No copyright stored; no message.

Publishing Status

Concept article: Publishing Status

Alerts

ALERT CONDITION	MESSAGE	IMPACT
No product-level **<PublishingStatus>**, and a **<ProductSupply>** lacks a **<MarketPublishingStatus>**	Product has no PublishingStatus and ProductSupply has no MarketPublishingStatus.	The ONIX file is rejected. Deliveries with **<NotificationType>** 05 (delete) are exempt.

Publishing Date

Concept article: Publishing Date

Alerts

ALERT CONDITION	MESSAGE	IMPACT
No sale start date at any level: no **<PublishingDate>** with role 01/02, no **<MarketDate>** with role 01/02, and no **<SupplyDate>** sales embargo	No publication or embargo date found in ONIX metadata.	The ***audiobook*** fails ***ingestion***.

Market Date

Concept article: Market Date

Alerts

ALERT CONDITION	MESSAGE	IMPACT
A **<MarketPublishingDetail>** contains a **<MarketDate>** but no **<MarketPublishingStatus>**	**<MarketPublishingStatus>** is mandatory in each occurrence of the **<MarketPublishingDetail>** composite	The ***audiobook*** fails ***ingestion***.

Sales Rights

Concept article: Sales Rights

Alerts

CONDITION	MESSAGE	IMPACT
<PublishingDetail> has no **<SalesRights>** and no **<ROWSalesRightsType>**	**<Product>** has no **<SalesRights>** or **<ROWSalesRightsType>** in <PublishingDetail>.	The ***audiobook*** fails ***ingestion***.

Price

Concept article: Price Acceptance

Alerts

ALERT CONDITION	MESSAGE	IMPACT
The evaluation moment is before the from date or after the until date	Price rejected — outside valid date range.	The **<Price>** composite contributes no price during Price Selection.

Currency

Concept article: Currency

Alerts

ALERT CONDITION	MESSAGE	IMPACT
<CurrencyCode> is not a code from list 96	**<CurrencyCode>** must be a valid ONIX code when present	The ***audiobook*** fails ***ingestion***.
A **<Price>** composite has a **<PriceAmount>** but no **<CurrencyCode>**, and no **<DefaultCurrencyCode>** is in the header	**<Price>** has **<PriceAmount>** but no **<CurrencyCode>** and no **<DefaultCurrencyCode>**.	The ***audiobook*** fails ***ingestion***.

Price Type

Concept article: Price Type

Alerts

CONDITION	MESSAGE	IMPACT
A priced **<Price>** composite has no **<PriceType>**, or a valid code other than `01`, `02`, `41`, `42`	**<Price>** rejected - **<PriceTypeCode>** or **<PriceTypeQualifier>** not accepted.	The **<Price>** composite contributes no price during Price Selection.
A priced **<Price>** composite has no **<PriceType>**, or a valid code other than `01`, `02`, `41`, `42`	**<Price>** rejected **<PriceTypeCode>** or **<PriceTypeQualifier>** not accepted.	The **<Price>** composite contributes no price during Price Selection.
A tax-inclusive price (`02`, `42`) has a **<Tax>** composite without a **<TaxableAmount>**	**<Tax>** composite is missing **<TaxableAmount>** on a tax-inclusive price.	The **<Price>** composite contributes no price during Price Selection.
A tax-inclusive price (`02`, `42`) has no **<Tax>** composite at all	**<Price>** rejected **<PriceTypeCode>** or **<PriceTypeQualifier>** not accepted.	The **<Price>** composite contributes no price during Price Selection.

Price Amount

Concept article: Price Amount

Alerts

ALERT CONDITION	MESSAGE	IMPACT
<TaxableAmount> is negative	**<TaxableAmount>** is negative.	The ***audiobook*** fails ***ingestion***.
<TaxableAmount> cannot be parsed as a number	**<TaxableAmount>** must contain a valid number when present	The ***audiobook*** fails ***ingestion***.
A tax-inclusive price has a **<Tax>** composite without a **<TaxableAmount>**	**<Tax>** composite is missing **<TaxableAmount>** on a tax-inclusive price.	The **<Price>** composite contributes no price during Price Selection.
A tax-inclusive price has no **<Tax>** composite at all	**<Price>** rejected - **<PriceTypeCode>** or **<PriceTypeQualifier>** not accepted.	The **<Price>** composite contributes no price during Price Selection.

Price Qualifier

Concept article: Price Qualifier

Alerts

ALERT CONDITION	MESSAGE	IMPACT
<PriceQualifier> is not a code from list 59	**<PriceQualifier>** must be a valid ONIX code when present	The ***audiobook*** fails ***ingestion***.
<PriceQualifier> is a valid code other than `00` or `05`	**<Price>** rejected - **<PriceTypeCode>** or **<PriceTypeQualifier>** not accepted.	The **<Price>** composite contributes no price during Price Selection.

Unpriced Item Type

Concept article: Unpriced Item Type

Alerts

ALERT CONDITION	MESSAGE	IMPACT
A **<Price>** contains both **<UnpricedItemType>** and **<PriceAmount>**	Cannot have **<PriceAmount>** and **<UnpricedItemType>**	The ***audiobook*** fails ***ingestion***.
A **<SupplyDetail>** contains both **<UnpricedItemType>** and **<Price>**	Each **<SupplyDetail>** composite must include either one or more prices, or a single **<UnpricedItemType>** element, but not both	The ***audiobook*** fails ***ingestion***.

A.2. Common ONIX Processing Rules

The rules in this section apply to all text-based fields, including title, description, contributors, and series name, unless a reference article specifies different behavior.

Text Processing

Spotify processes text extracted from ONIX elements in the following order:

1. **HTML entity decoding**: Named and numeric entities are decoded to their Unicode equivalents — for example, `&` becomes `&`, and `&#x2019;` becomes `’`.
2. **Whitespace normalization**: Leading and trailing whitespace is trimmed.

Language Resolution

The language assigned to a metadata element may affect how Spotify displays it based on each ***end user’s*** localization settings. This metadata language is distinct from the audiobook’s spoken language

Providing the correct language information helps Spotify categorize the audiobook appropriately and display its metadata in localized contexts.

For each ONIX element that supports a `language` attribute, Spotify resolves the metadata language using the first available source in the following priority order:

PRIORITY	SOURCE
1	The `language` attribute on the element itself, , such as **`<TitleText language="eng">`**.
2	The audiobook-level language from the first **`<Language>`** composite in **`<DescriptiveDetail>`** with a **`<LanguageRole>`** of 01 or 08.
3	**`<DefaultLanguageOfText>`** from the ONIX **`<Header>`**.
4	`und` (undetermined), when no language is available from the preceding sources.

Example

```
<ONIXMessage>
 <Header>
   <!-- Priority 3: Default language for all products in this message -->
   <DefaultLanguageOfText>eng</DefaultLanguageOfText>
   ...
 </Header>
```

```
  <Product>
    ...
    <DescriptiveDetail>
      <!-- Priority 2: Audiobook-level language -->
      <Language>
        <LanguageRole>01</LanguageRole>
        <LanguageCode>spa</LanguageCode>
      </Language>
      <TitleDetail>
        <TitleType>01</TitleType>
        <TitleElement>
          <TitleElementLevel>01</TitleElementLevel>
          <!-- Priority 1: Element-level language attribute -->
          <TitleText language="spa">Las vocales piratas</TitleText>
        </TitleElement>
      </TitleDetail>
      ...
    </DescriptiveDetail>
    ...
  </Product>
</ONIXMessage>
```

Undetermined Language (und)

Spotify does not reject an audiobook solely because its language is missing. Instead, Spotify assigns und (undetermined). This value may limit language-based discovery and localized display, and it is also cascaded onto other metadata fields like title, subtitle, and description tags.

To support accurate categorization, Language ***should*** be provided in a **`<Language>`** composite with a **`<LanguageRole>`**.

Accepted Codes

A list of accepted language codes based on a subset of codes found in [ONIX code list 22]

IDENTIFIERS	ACCEPTED CODE VALUES
Language Role	01, 08

Embedded Metadata

Spotify does not read metadata embedded in audiobook files. All metadata ***must*** be delivered through ONIX.

Metadata as Input

Metadata delivered through ONIX serves as input to Spotify's ingestion and catalog pipeline. The final values displayed to end users may differ from the submitted values. This specification describes the

metadata Spotify reads and processes from an ONIX feed; it does not guarantee how that metadata will appear across every Spotify experience.

Date Resolution

<Date> values are always resolved to a specific point in time, even when the original ONIX value contains only a date.

For each ONIX element that carries a **<Date>** element, Spotify resolves the value using the **<DateFormat>** code provided. When no **<DateFormat>** is specified, `00` is assumed. When no timezone is specified, UTC is assumed. [ONIX code list 55]

FORMAT CODE	RESOLVED AS
00	Midnight UTC is assumed (e.g., `20250115` becomes `2025-01-15T00:00:00Z`)
13	Hours and minutes are preserved as provided, and seconds are assumed 00
14	Hours, minutes, and seconds are preserved as provided

For validation messages and edge cases, see Validation : Dates

A.3. Example Audiobook

```
<?xml version="1.0" encoding="UTF-8"?>
<ONIXMessage release="3.0" xmlns="http://ns.editeur.org/onix/3.0/reference">
 <Header>
   <Sender>
     <SenderIdentifier>
       <SenderIDType>01</SenderIDType>
       <IDValue>example-publisher</IDValue>
     </SenderIdentifier>
     <SenderName>Example Publisher Inc.</SenderName> <!-- primary identifier for the
sending organization -->
   </Sender>
   <SentDateTime>20250615T120000</SentDateTime> <!-- timestamp of this ONIX file's
creation -->
 </Header>
 <Product>
   <RecordReference>9781234567890</RecordReference>
   <NotificationType>03</NotificationType>
   <ProductIdentifier>
     <ProductIDType>03</ProductIDType> <!-- type of product identifier GTIN -->
     <IDValue>9781234567890</IDValue> <!-- product identifier -->
   </ProductIdentifier>
   <ProductIdentifier>
     <ProductIDType>15</ProductIDType> <!-- type of product identifier ISBN -->
     <IDValue>9781234567890</IDValue>
   </ProductIdentifier>
   <DescriptiveDetail>
     <ProductComposition>00</ProductComposition>
     <ProductForm>AJ</ProductForm> <!-- this ONIX is for an audiobook -->
     <TitleDetail>
       <TitleType>01</TitleType>
       <TitleElement>
         <TitleElementLevel>01</TitleElementLevel>
         <TitleText language="eng">The Audiobook Example</TitleText> <!-- title of
audiobook, language of title is English -->
         <Subtitle language="eng">An Example of ONIX</Subtitle> <!-- subtitle of
audiobook, language of subtitle is English -->
       </TitleElement>
     </TitleDetail>
     <Contributor>
       <SequenceNumber>1</SequenceNumber>
       <ContributorRole>A01</ContributorRole> <!-- contributor is an author -->
       <PersonName>John Smith</PersonName> <!-- contributor name -->
     </Contributor>
     <Contributor>
       <SequenceNumber>2</SequenceNumber>
       <ContributorRole>E07</ContributorRole> <!-- contributor is a narrator -->
       <PersonName>Jane Doe</PersonName>
     </Contributor>
     <Contributor>
       <SequenceNumber>3</SequenceNumber>
       <ContributorRole>B06</ContributorRole> <!-- contributor is a translator -->
       <PersonName>Alex Johnson</PersonName>
```

```
      </Contributor>
      <Language>
        <LanguageRole>01</LanguageRole> <!-- language of audiobook text -->
        <LanguageCode>eng</LanguageCode> <!-- language of the audiobook, English -->
      </Language>
      <Subject>
        <SubjectSchemeIdentifier>10</SubjectSchemeIdentifier> <!-- BISAC subject category
-->
        <SubjectCode>FIC028000</SubjectCode>
      </Subject>
      <Subject>
        <SubjectSchemeIdentifier>93</SubjectSchemeIdentifier> <!-- Thema subject category
-->
        <SubjectCode>FL</SubjectCode>
      </Subject>
      <Subject>
        <SubjectSchemeIdentifier>29</SubjectSchemeIdentifier> <!-- CLIL classification
theme -->
        <SubjectCode>3442</SubjectCode>
      </Subject>
      <Collection>
        <CollectionType>10</CollectionType> <!-- this audiobook is part of a series -->
        <TitleDetail>
          <TitleType>01</TitleType>
          <TitleElement>
            <TitleElementLevel>02</TitleElementLevel>
            <PartNumber>1</PartNumber> <!-- position of the audiobook in the series -->
            <TitleText>The Example Series</TitleText> <!-- series name -->
          </TitleElement>
        </TitleDetail>
      </Collection>
      <EditionType>UBR</EditionType> <!-- this audiobook is unabridged -->
      <Extent>
        <ExtentType>09</ExtentType>
        <ExtentValue>10680</ExtentValue> <!-- duration of the audiobook in seconds -->
        <ExtentUnit>06</ExtentUnit>
      </Extent>
    </DescriptiveDetail>
    <CollateralDetail>
      <TextContent> <!-- this composite represents the description -->
        <TextType>03</TextType>
        <ContentAudience>00</ContentAudience>
        <Text language="eng">An example description that would be displayed.</Text> <!--
description in English -->
      </TextContent>
    </CollateralDetail>
    <ContentDetail> <!-- the start of Chapter Level Metadata -->
      <ContentItem> <!-- the first track of the audiobook -->
        <LevelSequenceNumber>1</LevelSequenceNumber>
        <TitleDetail>
          <TitleType>01</TitleType>
          <TitleElement>
            <TitleElementLevel>01</TitleElementLevel>
            <TitleText>Introduction</TitleText> <!-- the track's title -->
          </TitleElement>
        </TitleDetail>
        <AVItem>
```

```
          <AVItemType>01</AVItemType>
          <AVDuration>0000500</AVDuration>
        </AVItem>
      </ContentItem>
      <ContentItem> <!-- the second track of the audiobook -->
        <LevelSequenceNumber>2</LevelSequenceNumber>
        <TitleDetail>
          <TitleType>01</TitleType>
          <TitleElement>
            <TitleElementLevel>01</TitleElementLevel>
            <TitleText>Chapter 1: The Example Chapter Name</TitleText>
          </TitleElement>
        </TitleDetail>
        <AVItem>
          <AVItemType>01</AVItemType>
          <AVDuration>0025000</AVDuration>
        </AVItem>
      </ContentItem>
      <ContentItem> <!-- the third track of the audiobook -->
        <LevelSequenceNumber>3</LevelSequenceNumber>
        <TitleDetail>
          <TitleType>01</TitleType>
          <TitleElement>
            <TitleElementLevel>01</TitleElementLevel>
            <TitleText>Closing Credit</TitleText>
          </TitleElement>
        </TitleDetail>
        <AVItem>
          <AVItemType>01</AVItemType>
          <AVDuration>0000300</AVDuration>
        </AVItem>
      </ContentItem>
    </ContentDetail>
    <PublishingDetail>
      <Publisher>
        <PublishingRole>01</PublishingRole>
        <PublisherName>Example Publisher Inc.</PublisherName> <!-- publisher of this
audiobook -->
      </Publisher>
      <CopyrightStatement> <!-- text copyright -->
        <CopyrightType>C</CopyrightType>
        <CopyrightYear>2025</CopyrightYear>
        <CopyrightOwner>
          <PersonName>John Smith</PersonName>
        </CopyrightOwner>
      </CopyrightStatement>
      <CopyrightStatement> <!-- audio copyright -->
        <CopyrightType>P</CopyrightType>
        <CopyrightYear>2025</CopyrightYear>
        <CopyrightOwner>
          <CorporateName>Example Publisher Inc.</CorporateName>
        </CopyrightOwner>
      </CopyrightStatement>
      <PublishingStatus>04</PublishingStatus>
      <PublishingDate>
        <PublishingDateRole>01</PublishingDateRole>
        <Date>20250115</Date> <!-- the date of the full release -->
```

```
      </PublishingDate>
      <SalesRights>
        <SalesRightsType>01</SalesRightsType>
        <Territory>
          <CountriesIncluded>US GB CA AU</CountriesIncluded> <!-- title is allowed to be
available in these countries -->
        </Territory>
      </SalesRights>
    </PublishingDetail>
    <ProductSupply>
      <Market>
        <Territory>
          <CountriesIncluded>US GB CA AU</CountriesIncluded> <!-- title is allowed to be
active in these markets -->
        </Territory>
      </Market>
      <MarketPublishingDetail>
        <MarketPublishingStatus>04</MarketPublishingStatus>
      </MarketPublishingDetail>
      <SupplyDetail>
        <Supplier>
          <SupplierRole>00</SupplierRole>
          <SupplierName>Example Supplier</SupplierName>
        </Supplier>
        <ProductAvailability>20</ProductAvailability>
        <Price>
          <PriceType>01</PriceType>
          <PriceAmount>24.99</PriceAmount> <!-- US dollar retail price -->
          <CurrencyCode>USD</CurrencyCode>
          <Territory>
            <CountriesIncluded>US</CountriesIncluded> <!-- for sale in US for $24.99 -->
          </Territory>
        </Price>
        <Price>
          <PriceType>01</PriceType>
          <PriceAmount>19.99</PriceAmount> <!-- British Pound retail price -->
          <CurrencyCode>GBP</CurrencyCode>
          <Territory>
            <CountriesIncluded>GB</CountriesIncluded> <!-- for sale in GB for £19.99 -->
          </Territory>
        </Price>
      </SupplyDetail>
    </ProductSupply>
  </Product>
</ONIXMessage>
```