



Technical Handbook for Release Deliveries

Document Number: **Hand-0684**

Date of initial Version: **2011-11-16**

Date of current version: **2011-12-15**

Version: **1.0**

1. Introduction

This document is a DDEX Technical Handbook, aimed at explaining how to implement and use the DDEX standards to technical implementers as well as operational personnel.

This Technical Handbook will cover the standards and processes to deliver Releases and Products from record companies to online retailers/digital service providers.

This DDEX Technical Handbook for Release Deliveries covers the following areas:

1. Introduction.....	2
2. NewReleaseMessage.....	3
2.1. Message Header.....	4
2.2. Update Indicator.....	4
2.3. Resource List.....	4
2.4. Release List.....	5
2.5. Deal List.....	5
3. Choreography to communicate NewReleaseMessages and Resources.....	6
4. Starting an Implementation.....	6
4.1. Introduction — An Implementation Strategy.....	6
4.2. Technical Information Gathering.....	7
4.3. Choosing a Delivery Choreography.....	7
4.4. Metadata “Conformance”.....	7
4.5. Choreography “Conformance”.....	8
4.6. Going Live.....	8
5. Delivery of Metadata and Resources.....	9
5.1. Product Deliveries using FTP.....	9
5.2. Product Deliveries using Web Services.....	9
5.3. Asymmetric Web Service Architecture.....	12
6. Definitions and Terminology.....	16

2. NewReleaseMessage

The DDEX message to communicate details about a release and its availability is the NewReleaseMessage defined in the Electronic Release Notification Message Suite Standard (formerly referred to as “ERN”).

The NewReleaseMessage is the industry standard mechanism to allow content owners to inform DSPs about new releases that are available for distribution, and the terms and conditions under which releases can be made available. It includes complete metadata about the release and all resources contained within the release. In addition, it includes deals that describe when, where and how the release can be made available. The standard is available from <http://ddex.net/release-delivery-0>, profiles of the Electronic Release Notification Message Suite Standard are available from the same location.

The NewReleaseMessage is a XML specification defined and maintained by DDEX standards organisation. The NewReleaseMessage consists of five individual sections of message and metadata elements that are interrelated. All together they make up the NewReleaseMessage. A NewReleaseMessage will contain the details for an individual product (e.g. album, single, bundle) and each of its components (e.g. tracks). The message will not contain multiple products, related or non-related.

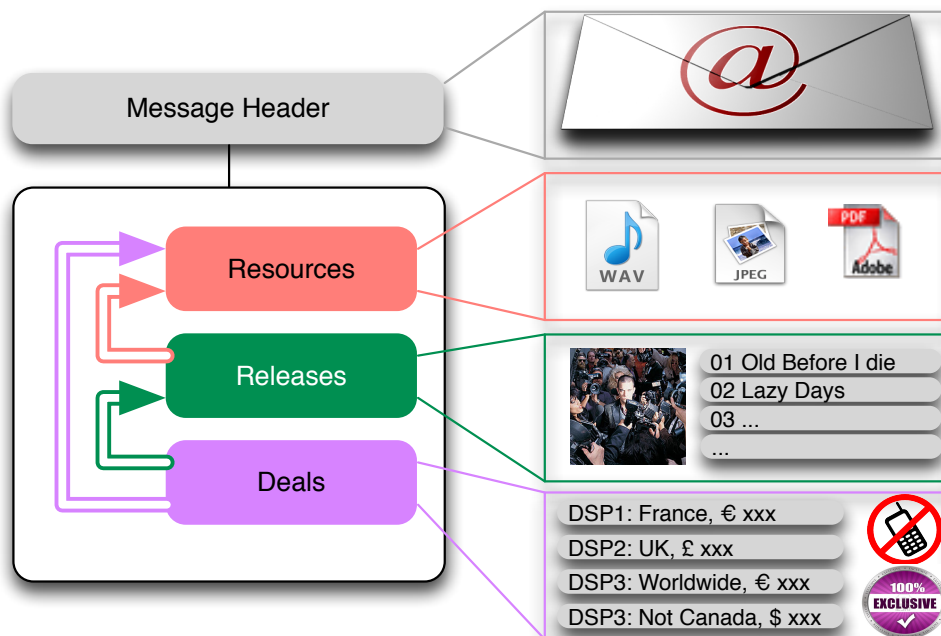


Figure 1: Overview of the NewReleaseMessage

The NewReleaseMessage consists of these sections that are described below:

- Message Header
- Update Indicator
- Resource List
- Release List
- Deal List

2.1. Message Header

The Message Header indicates the sender and recipient of the NewReleaseMessage Message.

The sender and recipient are each defined by a unique DDEX Party ID (DPID). The Message Header also provides a creation date which is a timestamp of when the message was created. DPIDs can be obtained at URL.

2.2. Update Indicator

The Update Indicator defines whether the NewReleaseMessage message is being received for the first time or is a subsequent update message. “Original Message” is the value for first time receiving and “Update Message” is the value for updates to indicate an update to metadata and/or deals.

For a takedown, the Update Indicator will still use a value of “Update Message”, but the Deal will indicate the details of the takedown.

2.3. Resource List

The Resource List provides details of the different assets that make up the entire release. Typical resources are sound recordings, videos and images. Each resource will have a unique reference anchor within the message (e.g. A1, A2, A3) which corresponds to the track number (e.g. track 1, 2, 3). The resource references will be referenced later in the Release section. The metadata defined in the Resource includes ISRC, artist, title and contributors.

Also contained in the Resource section is the Technical Resource Details, which references the specific binary files with details including the codec type and bit rate being delivered to the partner. The Technical Resource Details will provide information about both the full length and preview clip binary files.

Artwork will be a resource listed after the track resources. Cover artwork will be identified as an image resource with a resource reference after the last track resource. For example, on a 10-track album resource reference A11 would be the image resource relating to the cover artwork.

2.4. Release List

The Release List section defines the different releases that make up the “product” being delivered. For example, a standard ten-track album will contain one album release and 10 track releases, for a total of 11 Releases. The album level release is represented by the release reference R0 and the tracks by release reference R1, R2, R3, etc. dependent upon their order in the track listing. For each Release, the resources (“tracks”) included in the respective release are listed along with their sequence. These resources are referenced back to resources in the Resource List. For example, the Release for Track 1 references the Resource identified by Resource Reference A1, Track 2 by Resource Reference A2, Track 3 by Resource Reference A3, etc.

The metadata defined for each Release includes one or more Release identifiers (ICPN, UPC, EAN, GRid, ISRC), artist, title, label, genre, parental warning and P&C credits. A Release does NOT contain any information regarding the business terms associated with its commercial exploitation and usage.

2.5. Deal List

The Deal List provides the availability of the Releases. It describes when, where and how the releases can be made available. The Deal List provides a list of Release Deals, each having a reference back to a Release and defining the deal terms. Each Release Deal describes the commercial terms and usage information for each available Release. This includes the territories the Release can be made available, commercial model, usage type, pricing and deal validity period for each release listed. The deal start date usually represents the Release Date when the release can be made available to consumers in the specified territory.

Each Release Deal references back to the relevant Release using Release References. For example, the Release Deal for the Album Release references the Release identified by Release Reference R0, Track 1 by Release Reference R1, Track 2 by Release Reference R2, etc.

Within a Deal, a pre-order date can be specified. The pre-order date defines when the Release can be made available for pre-order, but not fulfilled to the consumer until the Deal

start date. A pre-order preview date can also be specified. This defines if and when the preview clips can be made available to the consumer during the pre-order period.

Deals are also used to notify of Takedowns. A takedown is the blanket loss of all rights and can be applied to specific territories or all territories. A takedown implies that all previous deals are cancelled in the specified territories on the takedown deal start date.

3. Choreography to communicate NewReleaseMessages and Resources

DDEX defines two mechanisms to exchange DDEX messages between two business partners. As such it establishes a (reliable) “pipe” to transfer DDEX messages, other XML documents and/or files, including binaries containing, for instance, Releases or Resources.

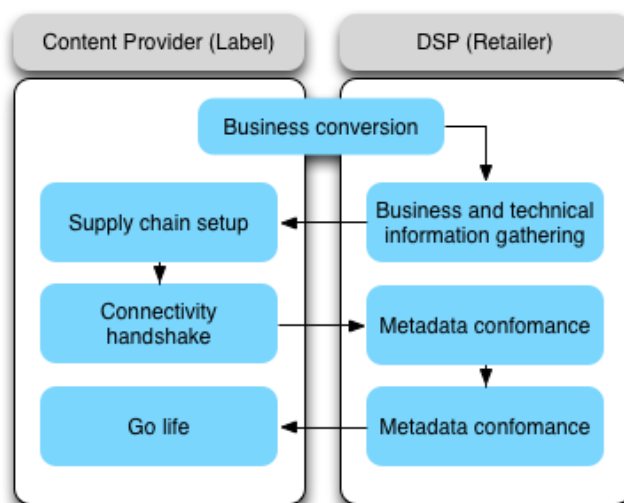
The entry-level message exchange protocol is based on FTP, the File Transfer Protocol. For parties who wish to implement a more interactive method of exchanging information with their business partners, this standard defines a second message exchange protocol using REST-like web services.

4. Starting an Implementation

4.1. Introduction — An Implementation Strategy

This section provides an introduction to the digital supply chain integration process between a content provider and a digital service provider (DSP) utilising the Digital Data Exchange (DDEX) standards, i.e.:

- Initial business and technical information gathering
- Supply chain handshake
- Content delivery “conformance”
 - Metadata as communicated in a NewReleaseMessages
 - Choreography conformance
- Going live



4.2. Technical Information Gathering

Before starting the technical supply chain's integration process, both, content provider and DSP operations personnel, need to gather the technical information and business requirements to configure their part of the supply chain. The information usually includes:

- DDEX implementation licence & DDEX Party ID (DPID) (To obtain both please go to URL);
- Populate Release and Resource information within the Release Profile;
- Populate Commercial terms within the Business Profile:
 - Release offering types;
 - Commercial and licensing models;
 - Pricing information;
 - Territories;
- Resource encode formats;
- Choice of the delivery choreography; and
- Server information and user account credentials.

4.3. Choosing a Delivery Choreography

After receiving the business and technical information, the content provider and DSP will configure the supply chain to accept/receive the content delivery and/or message communications.

The orchestration tests could include (s)ftp server and Web Servers communications, depending on the chosen choreography.

4.4. Metadata “Conformance”

During the integration phase, the content provider and DSP will need to conduct a “peer conformance procedure” where (i) the content owner determines if the DSP is able to receive and ingest the content according to the provided test packages and (ii) the DSP reviews the test packages from the content owner to verify the delivered packages complies with the standards.

The content provider will send a series of sample deliveries containing test products and business scenarios. These sample deliveries are intended to test the DSP's ability to receive/ingest content using the chosen delivery choreography, as well as their ability to correctly interpret the NewReleaseMessages.

- DDEX provides both content providers and DSPs with a consistent means of interpreting the metadata within NewReleaseMessages. During the integration phase, the content provider would typically send the DSP various XML messages to the partner, representing the following:
- Different Release types, e.g. single release, album release, etc.; (The most common Release types are defined the Release Profile Standard for Common Release Types, which is available from URL);
- Different Deal terms: single/multi-territories, multiple deal terms, price code/values, etc.; (The most common Deal types are defined the Business Profile Standard for Common Deal Types, which is available from URL);
- Different Business Scenarios; insert, update, revoke, takedown

The test messages, delivered during the conformance test phase, will have the IsTestFlag set to true to indicate that the messages are test messages and their content should not be made available to real consumers.

These messages ensure that the DSP's ingestion engine interprets the NewReleaseMessages accurately. During the Metadata conformance process, the content provider will need to verify the content and message ingestion results with each DSP based on their choice of choreography.

4.5. Choreography “Conformance”

Content owner and DSP will need to agree upon the best protocol and choreography for sending/receiving the metadata about the Releases as well as for receiving the Resource files that make up these Releases.

The different options are being described in subsequent Sections.

Note: DDEX is currently in the process of creating a Conformance standard that will provide information on how to test conformance of a company's processes.

4.6. Going Live

After the content provider and DSP passed the integration conformance tests, the content provider can “turn on” the digital feed in the supply chain to kick off the production deliveries, including the repertoire catalogue content and new releases.

5. Delivery of Metadata and Resources

As mentioned above DDEX supports multiple transport methods: FTP for delivery of the NewReleaseMessage and Resource files and web services.

5.1. Product Deliveries using FTP

File Transfer Protocol (FTP) is the standard protocol for transferring files to and from remote machines running FTP service. FTP offers less interactivity for the communication between labels and DSP.

Secure File Transfer Protocol (SFTP) is similar to FTP except that it encrypts both the command that execute the file transfer, as well as the data being transferred, whereby reducing the chances of eavesdropping during transfers.

Unique upload and download folders should be created according to the choreography standard. The standard is available from URL.

Messages should be ingested in order of folder timestamp.

Incoming and outgoing files should follow a standardised, descriptive naming convention, including version and/or timestamp. Such a standard reduces the possibility of processing the same file multiple times. These are also defined in the aforementioned standard.

5.2. Product Deliveries using Web Services

DDEX provides the ability to use web services for delivery of the metadata and also for communication between partners. There are many useful calls that cover the following capabilities:

- Well defined messaging interface between the sender's and receiver's system without interfering their autonomy
- Provide high visibility of the status of products in the supply chain for both parties (e.g. information about upcoming releases in the sending party's part of the supply chain)
- Allow quick reaction on problems without the need of calling somebody or writing emails (especially useful when managing different time zones)
- Have an automated process without the need of human interaction for all the actions performed frequently
- More control of the choreography and content flow for the receiving party

The Release Delivery Choreography Standard (see URL) contains detailed description of the entire choreography between involved parties. This includes all messaging definitions of the various calls. Note: A call always consists of a pair of messages, a request message and a response message.

The calls can be implemented as asymmetric or symmetric calls, both have advantages and disadvantages. Note: Some calls only make sense in the symmetric context, thus are not supported for the asymmetric choreography. Please refer to the tables below for a detailed list of supported WebService calls.

In the referenced documentation one can find the set of calls. One might think: Do I really need them all? There is a very clear answer to that: If you want to take the advantage of having full visibility and control over your supply chain and the content it received then, yes you should make use of all of them. But it is better to start with the basic ones and build up the remaining later on, than ignore all calls all together. We have grouped the choreography options and messages used into the following four groups. Each group provides an additional level of visibility and control to your supply chain.

REST

DDEX uses REST-like web services – should you need help in implementing this language then reach out to the DDEX community as some members already have implemented the WebServices choreography and might be able to help you there.

Each of the messages can be validated with an XML Schema files (XSDs, they are available from <http://ddex.net/xml>), making it easy to filter invalid messages from the start. DDEX highly recommends making use of the XSDs to prevent problems at later stages in your process. Not only to validate the messages you receive, but also for the messages you send.

Hint: DDEX works with namespaces in the XSDs, so if the messages you generate look good, but still fail, that is one of the first things to check.

Different programming languages and tools can be used to simplify the coding of the messages and also the processing of them, but make sure you really know your tool. The schemas DDEX is using are very complex, making use of sophisticated XML schema constructs like unions, inheritance, choices etc. This approach makes the XML schemas very smart and easy to read, but demand from the tools that they are able to cope with these complex requests. Experience gained from other implementers we have the following easy tips:

- Use XML schema mapping tools only if you have expert knowledge or have the time and skills to build up expert knowledge. You will need it to work around its quirks.
- If you get stuck, consider simplifying the original XML schema only for your schema-mapping tool. E.g. for inheritance/extension just copy the inherited elements.
- Exclude parts from the mapping completely and manually parse them instead. E.g. set “NewReleaseMessage” to “xsd:any” and parse the “xsd:any” elements with SAX/DOM/something.
- As with any good coding standard, whenever you change the original DDEX schema, document what you have done. Otherwise it is likely you will forget which changes you made and have a hard time when you have to switch to a new DDEX version, because it is likely you will have to redo the changes in the new schema again to make your process work again.

Part of the choreography is also sending acknowledgements to confirm the receipt of a sender’s message. This part in the response of a call provides the possibility to send some information about the status of the received message, e.g. giving detailed error information.

Hint: DDEX has not created a list of valid error messages so far, nor has it defined a special structure what they should look like. The group is waiting for more members to implement a Web Service solution and find out the most common errors. Once done, a list will be provided to ease up the handling and harmonise the error messages.

Web Service Messages

There are messages to request new deliveries, to check the status of the product in the partners supply chain, to request reports, and many more.

All these messages contain a common part: the header. This identifies the sender and the recipient of the message, but also the message itself, for later reference (e.g. in the acknowledgment) and the type of WebService choreography chosen (symmetric or asymmetric). The priority of the message can be set here. Make sure not to send high priority requests all the time; this could make it impossible for your partner to identify high priority messages from such with standard priority. Additionally some optional values can be set, like the hash sum of the message, for example. To identify the order of the messages, in case there are multiple messages of the same type, then the message creation date is also incorporated (including time zones).

Hint: Resource files and also reports will not be messaged as WebService calls, they will be loaded to a defined location e.g. to the (s)ftp server. These files are excluded from

transfers via WebServices, because of their possible large size. All binary files and their individual location are referenced in the provided XMLs, though.

5.3. Asymmetric Web Service Architecture

In the asymmetric WebService messaging the communication is always initiated by only one partner, which is calling the WebService provided by the second partner.

- This leads to a high number of messages attempting to receive a changed response. This could lead into unnecessary messages, as instead of getting informed that there is something new, the partner needs to keep asking for news on a high frequency, maybe every 10 minutes to see if anything new has been posted.
- As mentioned previously already, some messages are not available in this choreography.

On the other hand, in this solution there is no need for both partners to host a server. There are calls providing information and calls requesting a specific action.

Asymmetric Web Service Calls	Basic Level		Full Level	
	Calling	Receiving	Calling	Receiving
DeliveryFrequencyChangeRequestCall	√		√	
RedeliveryRequestCall	√		√	
ReleaseAvailabilityRequestCall	√		√	
SupplyChainStatusCall	√		√	
OrderedReleasesInQueueRequestCall			√	
ReportRequestCall			√	
InformationAboutAvailableReleaseRequestCall			√	
ReleaseStatusInformationCall			√	

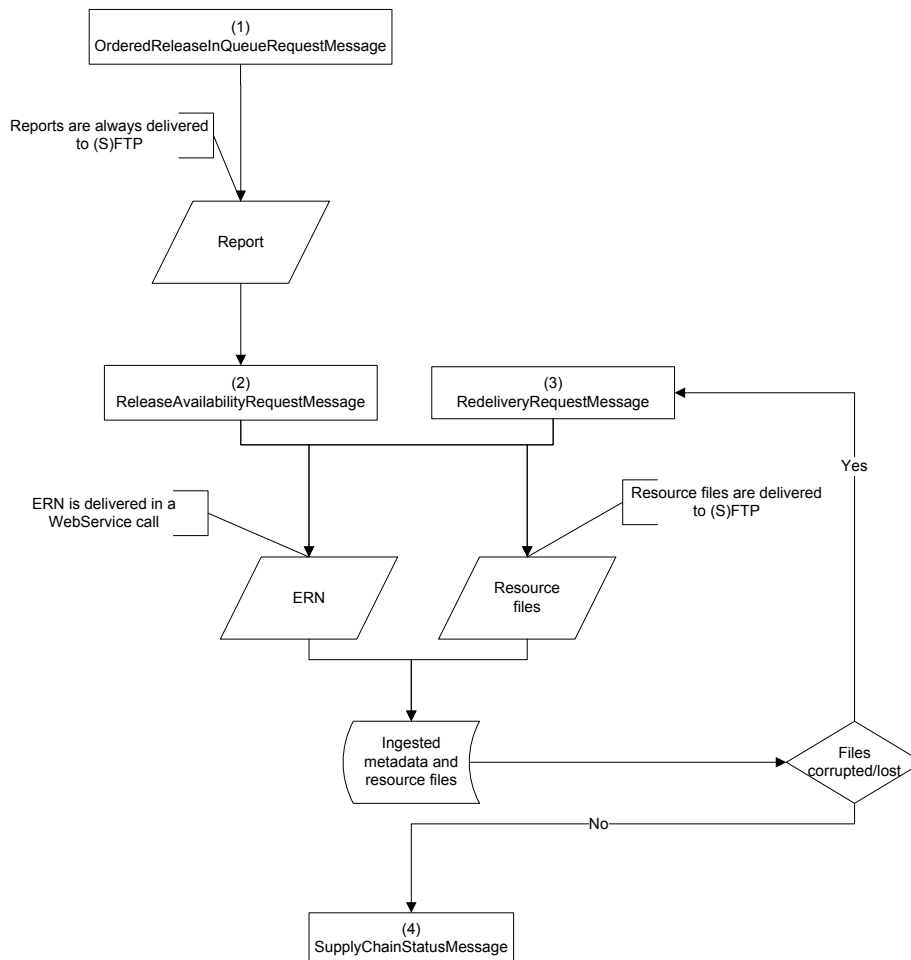


Figure 2: Lifecycle of a Release using asymmetric WebService calls

The ReleaseAvailabilityRequestMessage (2) can be used to request an ERN XML from a partner. Using a specific id inside the call a defined release can be requested. The response is either the ERN XML for the requested release, or (if it is not available for pick up yet) the status of this release in the message recipient’s supply chain. To be able to query for a specific release, the orders that are in queue for the partner have to be known. A list can be requested using the OrderedReleasesInQueueRequestMessage (1). The result of this call is a report, the feedback is performed asynchronously and the report is delivered to a predefined (S)FTP location. In the asymmetric WebService choreography, there is no call to inform the partner, that the report got created and is ready for pick up. The receiver of the report needs to check for its status. If something happened to the resource files of the ERN XML, a RedeliveryRequestMessage (3) can be used to request the release to be redelivered. There is no need to pick up the phone and ask somebody for the redelivery. After the files are ingested the SupplyChainStatusMessage (4) can be used to inform the partner of the status of the release in the system, e.g. everything ingested, release is up in the shop now... There are many more messages, for a detailed description and the full list please refer to the Release Delivery Choreography Standard available from <http://ddex.net/release-delivery-0>.

Symmetric Web Service Architecture

In the symmetric choreography both partners host a server and both are able to initiate the communication. This option greatly reduces the number of messages that are needed to perform the same actions and places more control in the hands of both partners.

Symmetric Web Service Calls	Basic Level		Full Level	
	Call	Receive	Call	Receive
DeliveryFrequencyChangeRequestCall	√		√	
RedeliveryRequestCall	√		√	
ReleaseAvailabilityRequestCall	√		√	
ReleaseAvailabilityCall		√		√
SupplyChainStatusCall	√		√	
ReleaseSupplyChainStatusRequestCall		√		√
OrderedReleasesInQueueRequestCall			√	
ReportRequestCall			√	
ReportDeliveryCall				√
InformationAboutAvailableReleaseRequestCall			√	
ReleaseStatusInformationCall			√	
ReleaseStatusRequestCall				√

Also the variety of calls that can be made is bigger.

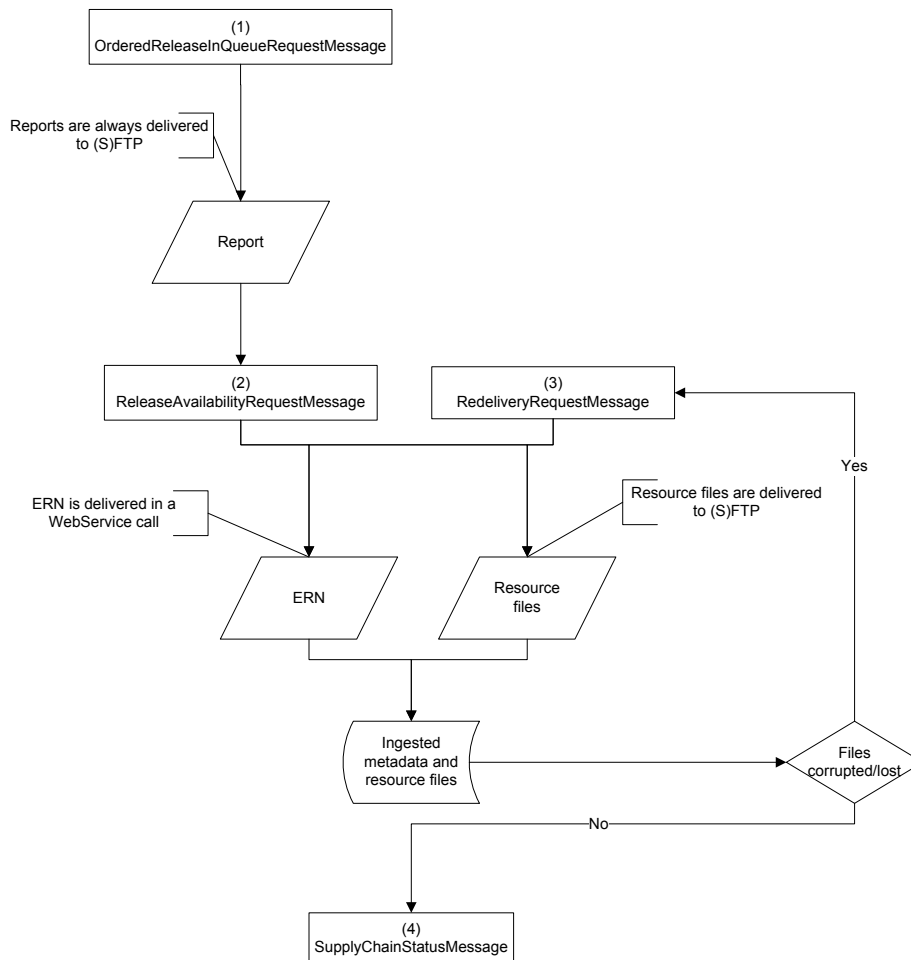


Figure 3: Lifecycle of a Release using symmetric WebService calls

Instead of waiting for a report to be delivered after using the OrderedReleaseInQueueRequestMessage (7), the partner can be informed immediately after the report is generated, using the ReportDeliveryMessage (6).

As soon as a new release is available for a partner the ReleaseAvailabilityMessage (1) is used to inform. Requesting releases in a specific order is possible. This can be used if there are high priority products, which the partner would like to receive prior to others, for example. In the asymmetric case, there is no way for the provider of releases to request the status of a release in the recipient’s supply chain in an automated way. The content provider can only wait until the SupplyChainStatusMessage arrives from the content recipient. In the symmetric choreography the ReleaseSupplyChainStatusRequestMessage(4) allows to actively requests the status of the product in the recipient’s supply chain. Again, there are a many more messages and for a detailed description and the full list of messages available please refer to the Release Delivery Choreography Standard available from <http://ddex.net/release-delivery-0..>

6. Definitions and Terminology

Aggregator — A company that sends or receives DDEX messages on behalf of other companies. Aggregators enable smaller companies to participate in the DDEX ecosystem.

Deal — A Deal is what transforms a Release into a product. A Deal states how a Release may be made available to consumers.

DSP — Digital Service Provider; a company that receives and/or distributes content

FTP — File Transfer Protocol; A protocol to exchange files (e.g. XML files or Resource files) via the internet.

Product — a collection of resources, combined with a deal that is marketed or sold as a commodity

Profile — a subset of a larger standard that defines how to use that standard in a specific way for a specific use case

Release — A collection of one or more Resources that, together, make a tradable collection.

Resource — The individual assets that make up a Release. Typical Resources are sound recordings, video clips and cover art images.

Web Service — A modern set of web technologies that allow small pieces of information, typically in the form of XML files, to be exchanged. Augmented with FTP they can be used to communicate Releases along the music supply chain.

XML — eXtensible Markup Language; a set of rules for encoding documents in machine-readable form.