Product Deliveries using Web Services

Introduction

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 and 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, 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.

> 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.

> 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).

> 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.