DMS General System Guid

Functional guide to DMS in general

UDVIKLINGS OG FORENKLINGS STYRELSEN

Table of contents

Introduction

1

This guide details the functionality of the new Declaration Management System – DMS in general. The target group for this system guide is developers responsible for developing a system-to-system (S2S) integration from their own customs clearance system to DMS System-to-System.

The aim of this document is to provide an understanding of the technical setup around a system-to-system integration. This System Guide explains the general message flows and functions that can be carried out with DMS. The appendix contains various tables, that are not specific to any of the customs regimes (Export/Import/Transit).

For details on how to establish a connection to DMS, consult the DMS Connectivity Guide (found on Danish Customs and Tax Administration's GitHub) instead, as this describes how to establish connection to the AS4-gateway after signing up for the system.

This document will be enhanced continuously. So far, the document covers functionality of DMS in general including shared functionality between DMS Import, DMS Export and DMS Transit. Functionality specific to Import/Export/Transit can be found in their respective guides.

This guide often references the following guides for more detailed information: DMS Import/Export System Guide, DMS Transit System Guide, DMS Notification Guide.

Technical overview

2

2.1 Relation between DMS System-to-System and DMS Online

DMS can be accessed either via a solution called DMS System-to-System (S2S), where declarations are submitted through the AS4-gateway, or via the systems online User Interface (UI) called DMS Online.

In principle, all system functions can be managed through both access points, UI or S2S. If using DMS System-to-System, the recommendation is to use the UI only to look up information to avoid the risk of mismatch of data between own backend and DMS, e.g., if a declaration is lodged via S2S integration but amended via the UI, your backend will not know which data was changed via the UI.

2.2 System overview

DMS has two types of actions: One for submitting declarations, additional messages, and IE-messages and one for requesting notifications, which are statuses of a declaration.

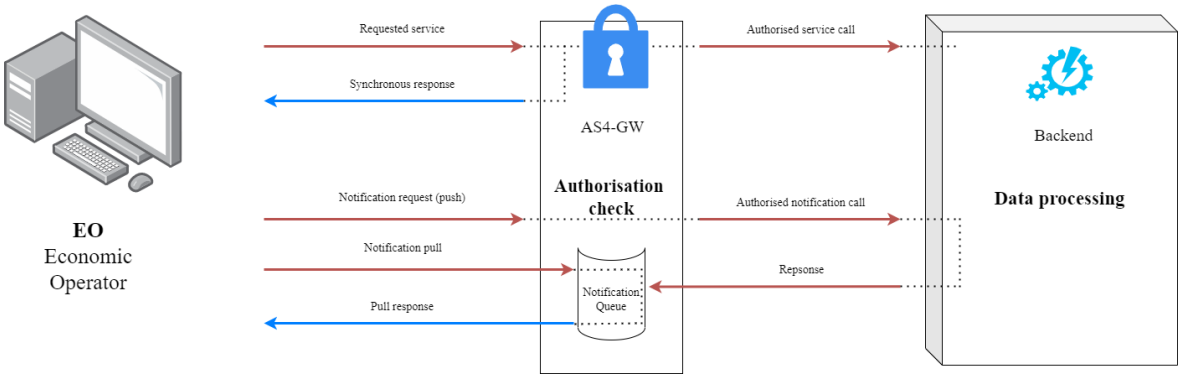

Figure 2-1– System overview

When lodging a declaration, additional message, or IE-message (requested service), the desired action is called. An illustration of this can be seen on Figure 2-1. In the AS4-gateway, the submitted XML is syntax validated, and a synchronous response is returned. In case of a syntax error, the request is synchronously rejected. If the syntax validation is passed, the declaration or additional message is sent to the backend system, where semantic validation and further processing of the declaration, additional message or IE-message will be carried out.

To know if the declaration, additional message, or IE-message has been accepted or received and what its status is, another action – the notification action – needs to be called by the user. When called with a specific set of parameters, the notification request returns all notifications available matching those request parameters.

2.3 Services and endpoints

All services and actions are reached through the AS4-gateway from the same endpoint, where the AS4-header indicates which service and action to call.
See details on endpoint and AS4-header in the DMS Connectivity Guide.
A description of all services and actions for each domain can be found in their respective system guides under “AS4 Services” (Transit System Guide section 6.3, Import/Export System Guide section 8.4).

The endpoints are:

Environment	Hostname	Port
UFE	secureftpgatewaytest.skat.dk	6384
TFE	secureftpgatewaytest.skat.dk	6384
Prod	secureftpgateway.skat.dk	6384

Table 2-1 - Endpoints

This information is combined with details of the company, creating a complete endpoint, e.g.:

https://secureftpgatewaytest.skat.dk:6384/exchange/CVR_{CVR}_UI_{UUID}

Notifications

3

This section describes how notifications (or declaration statuses) are managed in general.

After a declaration is submitted, it passes through the system in different states that describe where the declaration is in the customs process. The system communicates the state of a declaration through notifications.

Receiving notifications is a multi-part process. The parts of the process are **Pushing notification requests** and **Pulling notification responses**. Notification requests should be pushed with a given time interval and pulls of notification responses from the AS4-gateway queue should be interwoven in between – an example of how to work with notification pushes and pulls is described in section 3.1.

The notifications function on a **pagination** system, which means that only a limited number of notifications can be requested at a time. If the total number of available notifications exceeds the requested amount, multiple pages need to be requested.

A notification push contains the parameters “submitterId”, “dateFrom”, “dateTo”, “page” and “size” (see Table 3-1), which describe the time interval within which notifications should be retrieved, what page is wanted in the time interval and the number of notifications return per page. Note that the **time interval may not exceed 48 hours**. Upon pushing a notification request, a synchronous response will be elicited by the AS4-gateway in the form of an ‘OK’ response or an error.

When a push is performed successfully, the AS4-gateway will asynchronously retrieve a response from DMS and place it in a queue within the AS4-gateway from which a pull can then be performed. The pull will result in a synchronous response. The response will contain the notifications requested in the push, and the total amount of notifications in the time range.

It is important to continue pulling notifications from the AS4-gateway until the queue is empty, as not all notifications are generated from a request for specific notifications.

A notification holds information on which declaration the notification relates to, the state of the declaration, customs position response in relation to additional message requests, error codes (if rejected), etc.

In the DMS Notification Guide and the DMS Transit System Guide (section 4.2) you can find lists and descriptions of each notification type, how to read it from the notification response, and other relevant information. For information on which notifications to expect from the different declarations and additional message flows, see the DMS Import/Export System Guide section 8.1 - 8.3 and the DMS Transit System Guide section 6.1 - 6.2.

Please note in section 3.1.4 that all responses to a notification request are placed on - and therefore must be pulled from - the same Message Partition Channel (queue).

3.1 Notification Request Design Recommendation

Notifications are requested in time intervals. We recommend requesting all notifications that occurred in the last 7 minutes every 5 minutes, see the simplified example below. With the suggested pattern, an overlap is created to mitigate the possibility of missing notifications, therefore it is important to keep track of which notifications have already been received. Notifications can be uniquely identified by a NotificationSID element, which is present within the notification XML. This is described in the DMS Notifications Guide under section 2.2.

Action #	Time	Push	Pull	Sync Pull Response	Time range	Page	Size	Comment
1	12:00	1			11:53 - 12:00	0	500	A request is sent with a 7-minute time range, starting at page 0.
2	12:02		1					A pull is performed at 12:02.
3	12:02			1				Notification response is received at 12:02 containing 90 notifications from 11:53 to 12:00. As there are only 90 notifications in total, it's a single page.
4	12:05	2			11:58 - 12:05	0	500	A request is sent with a 7-minute time range, starting at page 0.
5	12:07		2					A pull is performed at 12:07.
6	12:07			2				Notification response is received at 12:07 containing 500 of a total of 590 notifications from 11:58 to 12:05. As there are 590 notifications in total, there is 2 pages that has to be requested in total.
7	12:07	3			11:58 - 12:05	1	500	A request is sent with the same 7-minute time range as in **action #4** but now requesting page 1 instead of page 0.
8	12:09		3					A pull is performed at 12:09.
9	12:09			3				Notification response is received at 12:09 containing the remaining 90 of the 590 notifications from 11:58 to 12:05. As there are 590 notifications in total, there are 2 pages. As we requested the second page in **action 7**, all notifications have been pulled successfully.

Table 3-1 - Notification Request Design (seconds are omitted for simplicity)

The notification system is based upon a pagination system. This allows for a more controlled flow of how many notifications are requested and delivered at once. This should help prevent receiving too much information at once.

The flow diagram below shows how the notification functions with the implemented pagination.

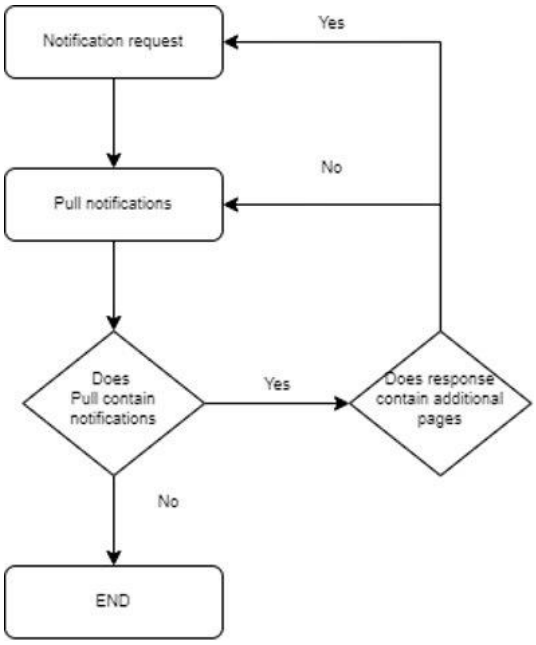

Figure 3-1 - Flowchart for requesting and pulling notifications

At the start of the flow, a notification request shall be made requesting **page 0** in the timeframe for which notifications are wanted, see chapter 3.1.1.

Then pull the notifications from the queue, as seen in the flow diagram. Check if the response contains any notifications, if there is none, then the flow is over, and the given timeframe is empty. If the response contains any notifications, check if it contains additional pages. Below is a response containing 130 notifications `<TotalSize>130</TotalSize>` in the requested timeframe.

```
<NotificationResult>
    <TotalSize>130</TotalSize>
    <Notifications>
        <Notification>
            ....
        </Notification>
    </Notifications>
</NotificationResult>
```

Figure 3-2 - Example of pull with 130 notifications

The notifications can arrive in bundles, where each notification bundle is indicated by the `<Notifications> ..... </Notifications>` tags (notice `Notifications` is in plural) and can contain

multiple notifications indicated by multiple `<Notification> ..... </Notification>` tags (`Notification` in singular). This is also illustrated in the notification response shown above.

To check if it contains additional pages, use the following equation.

$$\text{Pages} = \lceil \frac{TotalSize}{size} \rceil$$

Here the size is the amount specified in the push request, if none specified the default value is 500. Then round the result up to the nearest whole number.

$$\text{Pages} = \lceil \frac{130}{500} \rceil$$

The number of pages = **`ceil(130/500)= 1`**, which means that 130 notifications will result in 1 page. **Note** that the pages in the pull request are 0-indexed, which means the page count starts at 0.

Example 1: If you have only 1 page available, the total amount of pages to request is 1, but in the request, it is written as 0.

Example 2: If you have 5 pages, the total amount of pages to request is 5, but in the request, you will have to request the pages from *page = 0* to (and including) *page = 4*.

If there are additional pages, request the next one at the start of the flow diagram, continue this behavior until all pages in the timeframe are requested and pulled. If there are no additional pages, pull notifications again, as shown in the flowchart on Figure 3-3, as there might be other messages on the queue.

3.1.1 Requesting notifications

When requesting a set of notifications, the values below can be specified in the message properties. MessageId of the request is contained in the pull response alongside the requested notifications. The MessageId is part of MessageInfo in the AS4-header. NB: the **dateTo** and **dateFrom** fields are filled out from the perspective that you are pulling **From** a past time **To** a present time. The time difference between **dateTo** and **dateFrom** may not exceed 48 hours.

Attribute	Value	Example
submitterId	The submitter ID described in section 6.1.2 in the Connectivity Guide	13123456
dateTo	must be described in YYYY-MM-DDTHH:MI:ss.SSS **in time zone UTC-0** May not differ more than 48 hours from **dateFrom**	2022-02-17T12:07:00.000

dateFrom	must be described in YYYY-MM-DDTHH:MI:ss.SSS **in time zone UTC-0** May not differ more than 48 hours from **dateTo**	2022-02-17T12:00:00.000
lang	The language of the response (**must be EN**)	EN
page	The page wanted in the request. **Default 0**	0
Size	The page size wanted – **Default 500** **The max size is 500**	500

Table 3-2 - Notification push request header

An XML example of the MessageProperties is shown below:

```
<eb3:MessageProperties>
    <eb3:Property name="size">500</eb3:Property>
    <eb3:Property name="lang">EN</eb3:Property>
    <eb3:Property name="page">0</eb3:Property>
    <eb3:Property name="submitterId">13116482</eb3:Property>
    <eb3:Property name="dateFrom">2023-08-22T12:00:00.000</eb3:Property>
    <eb3:Property name="dateTo">2023-08-22T12:07:00.000</eb3:Property>
</eb3:MessageProperties>
```

Figure 3-3 - Example of the MessageProperties element

3.1.1.1 Requesting import, export and transit notifications

In the DMS Import/Export System Guide under section 8.4 and in the DMS Transit System Guide under section 6.3 there is a list of services for sending in declarations and requesting notifications.

3.1.2 Requesting status using MRN

It is also possible to request the status of a specific declaration using its MRN. An MRN query request is sent via a push request (see 3.1.1) to the AS4 Gateway, and the response must be pulled the same way as notifications and error messages (see 3.1.4) from the standard MPC.

The request is however sent through a specific service, using the endpoint:
DMS.Shared.Declaration.GetStatus

Note: Due to using a difference service, AS4 will not respond with the 200 OK response you expect to receive upon submitting requests to the DMS.Export/Transit/Import services. In a successful scenario for the MRN service, you should only expect 1 response, containing the MRN status response.

The request is sent in the same way as a regular notification request, with the following key differences.

- The endpoint is *DMS.Shared.Declaration.GetStatus*
- The URL path must be "/exchange/shared";
  e.g: https://secureftpgateway.skat.dk:6384/exchange/shared
- The following attributes should be used:

Attribute	Value	Example
mrn	Must contain the MRN to query	24DKCYYKR9EOCPTEB5
regime	Must contain one of the valid regime codes, which the specified declaration belongs: "EX" for Export, "IM" for Import, "TRA" for Transit.	EX

Table 3-3 - MRN status request header

Full examples of a MRN status request, as well as examples of different possible responses can be seen on The Tax Administration's GitHub.

3.1.3 Requesting an EAD/TAD using MRN/LRN

It is possible to request an Export Accompanying Document (EAD) or a Transit Accompanying Document (TAD) for a declaration through AS4 Gateway. An EAD/TAD request is sent via a push request to the AS4 Gateway (see 3.1.1). If the message is received successfully by AS4 Gateway, it will return a receipt. The response must be pulled the same way notifications and error messages from the standard MPC are pulled (see 3.1.2). The response payload is either a PDF document containing the EAD/TAD or, in case of an error, an XML document describing the error. The request is sent through the endpoint via *DMS.Declaration.GetEad* or *DMS.Declaration.GetTad* respectively.

This request, unlike other requests, does not have a body as all request parameters are specified as query parameters. The request is sent in the same way as a regular notification request, with the following differences:

- The endpoints are *DMS.Declaration.GetEad* and *DMS.Declaration.GetTad*
- The URL path must be "/exchange/shared",
  e.g: https://secureftpgateway.skat.dk:6384/exchange/shared
- The following attributes should be used.
-

Element	GetEad	GetTad	Example
Property.mrn	MRN of the declaration	MRN of the declaration*	24DKCYYKR9EOCPTEB5

Property.lrn	N/A	LRN of the declaration*	EXLRNExample2024

Table 3-4 - EAD/TAD request header

(*) Either only the LRN or only the MRN should be set. If both are set, then the LRN will be ignored as MRN takes precedence.

Full examples of an EAD\TAD notification request, a synchronous response, and a pull response can be seen on The Tax Administration's GitHub.

3.1.3.1 Errors

If no declaration could be found for the given MRN/LRN, then an XML with the 404 error will be returned. See Figure 3-4.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<genericErrorDTO>
    <cause>
    </cause>
    <message>Api Gateway Returned: 404 NOT_FOUND</message>
    <timestamp/>
</genericErrorDTO>
```

Figure 3-4 - Example of 404 Error

If the MRN\LRN exists but the associated declaration was created by another user than the one which is used to make the push request for the EAD\TAD, then a place holder error is returned. The error is a No Respons Body Error and an example can be seen on Figure 3-5.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ErrorMessageDTO>
    <lrn></lrn>
    <ValidationResponse>
        <ValidationResults>
            <ValidationResultType>B2BERR</ValidationResultType>
            <ValidationResultText>No response body was returned from the Trader Portal. Response status:
                406. ebmsConversationId: placeholder. ebmsMessageId: c0afed27-6e8f-48f3-adea-dfa1e638f11d.
                transactionId: ci1717416528192_15295647_4</ValidationResultText>
        </ValidationResults>
    </ValidationResponse>
    <timestamp>2024-06-03T14:08:57</timestamp>
</ErrorMessageDTO>
```

Figure 3-5 - Example of No Response Body Error

3.1.4 AS4 Notification Pull Request Header

This section explains which information should be contained in the AS4 header when pulling notifications from a Message Partition Channel (MPC).

The MPC identifies the queue on the AS4-gateway that you will receive messages from. You are expected to change out "********" with your CVR number, e.g. "$CVR\_12345678$".

Note: Only one queue per environment exists in DMS (see below). This means that any response to a notification request, whether it is a request for import, export, or transit notifications, will be placed on the same queue.

Environment	Message Partition Channel
Production	urn:fdc:dk.skat.mft.DMS/response/CVR_********
TFE	urn:fdc:dk.skat.mft.DMS/response/CVR_********
UFE	urn:fdc:dk.skat.mft.DMS/response/CVR_********

Table 3-5 - Message Partition Channel from which to pull notifications

The following attributes must be provided when pulling:

Attribute	Value	Example
Message-Info.Timestamp	YYYY-MM-DDTHH:MI:ss.SSSZ	2021-01-19T15:24:37.376Z
MessageInfo.Mes-sageId	GUID	4874bbf7-33c0-49cb-8b98-ca399fccf34a
PullRequest.Prop-erty[mpc]	Message Partition Channel	urn:fdc:dk.skat.mft.DMS/response/ $CVR\_$ ********

Table 3-6 - Header contents of notification pull

A full XML example of the notification pull messaging header is shown below:

```
<eb3:Messaging xmlns:ns3="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
env:mustUnderstand="true" wsu:Id="id-9db14fd7-2780-4e41-9078-87ba0a8e112f">
    <eb3:SignalMessage>
        <eb3:MessageInfo>
            <eb3:Timestamp>2024-01-15T16:15:27.8546192</eb3:Timestamp>
            <eb3:MessageId>5408f821-13a8-4fdb-8943-a1498dcbafd3</eb3:MessageId>
        </eb3:MessageInfo>
        <eb3:PullRequest mpc="urn:fdc:dk.skat.mft.DMS/response/CVR_12345678"/>
    </eb3:SignalMessage>
</eb3:Messaging>
```

Figure 3-6 - Notification pull message XML example

3.1.5 AS4 Notification Pull Response

A notification pull **response** from the AS4-gateway consists of the SOAP envelope, and, in case the queue is not empty, an additional XML document containing the actual notifications. How to read the additional XML document with the notifications depends on the domain and is described in the DMS Import/Export and DMS Transit System Guides respectively. A full response to a pull request can be seen on The Tax Administration's GitHub.

The message property "*RefToOriginalMessageId*" seen below contains the MessageID of the message used to generate the response, such as the MessageID of a request of specific notifications (see 3.1.1). Therefore, *RefToOriginalMessageId* should be used to keep track of responses to messages.

```
<eb:MessageProperties>
    <eb:Property name="RefToOriginalMessageId">a46efff9-e56b-48fb-8439-
627843cce50b</eb:Property>
</eb:MessageProperties>
```

```
<eb:CollaborationInfo>
    <eb:ConversationId>placeholder</eb:ConversationId>
</eb:CollaborationInfo>
```

Figure 3-7 - Example of RefToOriginalMessageID & CollaborationInfo

It is possible to use the *"ConversationId"* property under the "CollaborationInfo" element to track a **series** of messages by grouping them into a conversation. There is no specific format for this id, it can be anything you find suitable for your use-case. Examples of messages that includes the ConversationId and its sibling elements can also be found on GitHub.

When the queue is empty, the AS4 gateway response contains only the soap envelope. A complete example of an empty queue response, including the full soap envelope is shown below:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
<soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <eb:Messaging xmlns:eb=http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/
soapenv:mustUnderstand="true">
        <eb:SignalMessage>
            <eb:MessageInfo>
                <eb:Timestamp>2022-06-16T08:13:17.207Z</eb:Timestamp>
                <eb:MessageId>20fc5a2f-8184-40ee-80b4-89825a538f6f@SKAT-MFT</eb:MessageId>
                <eb:RefToMessageId>3c3d0efd-3e33-4749-9315-74a887e31ed5</eb:RefToMessageId>
            </eb:MessageInfo>
            <eb:Error category="Communication" errorCode="EBMS:0006" origin="ebMS"
severity="warning" shortDescription="EmptyMessagePartitionChannel">
                <eb:Description xml:lang="en-US">No message for requested MPC
(urn:fdc:dk.skat.mft.DMS/import2/response)</eb:Description>
<eb:ErrorDetail>//SOAPHeader/eb:Messaging/eb:SignalMessage/eb:PullRequest</eb:ErrorDetail>
            </eb:Error>
        </eb:SignalMessage>
    </eb:Messaging>
</soapenv:Header>
<soapenv:Body/>
</soapenv:Envelope>
```

Figure 3-8 - Empty notification pull example

The error returned for an empty queue **does not indicate a problem** with the communication, but simply that the queue has been emptied. This is **expected behavior**.

3.2 AS4 Error Messages

In case there are any errors with AS4, there is a variety of error messages that you can receive from the queue.

3.2.1 No valid token could be retrieved

This happens when the AS4 gateway cannot retrieve a token for the user to access the Trader Portal. This could e.g., be caused by the user trying to send to an endpoint they are not created in. The solution to this is to make an incident with TOLDST Service Desk. They will refer the case to the correct team depending on the environment.

```
  <ValidationResults>
    <ValidationResultType>B2BERR</ValidationResultType>
    <ValidationResultText>Request could not be sent because no access
token could be retrieved </ValidationResultText>
  </ValidationResults>
```

Figure 3-9 - No valid token example

3.2.2 Connection timed out

This error occurs when the response time exceed the maximally allowed time limit. Wait for a while and try again. If the problem persist, create a case for the DMS Onboarding Team for TFE or reach out to TOLDST Service Desk for PROD.

```
  <ValidationResults>
    <ValidationResultType>B2BERR</ValidationResultType>
    <ValidationResultText>Connection timed out: connect</ValidationRe-
sultText>
  </ValidationResults>
```

Figure 3-10 - Connection timeout example

3.2.3 CVR-Number from Payload is not valid to use

You will receive this error if you have registered an SE-number, but you are using CVR in your payload. Try finding your onboarding documents, locate your SE-number, and include it in the payload. Alternatively, if the SE-number was registered by mistake, go to the certificate portal and remove the SE-number.

```
  <ValidationResults>
    <ValidationResultType>B2BERR</ValidationResultType>
    <ValidationResultText>CVR-number 12345678 from payload is not
valid to use, because a SE-number is registered with this CVR </Valida-
tionResultText>
  </ValidationResults>
```

Figure 3-11 - CVR-number from payload is not valid to use example

3.2.4 SE-Number from Payload is not valid

You will get this error if you are using a wrong SE-number. Make sure you are using the correct SE-number and confirm that it is the one registered to the certificate. Note: If you have the same SE-number and CVR number, do not fill out the SE-number in the certificate portal. Doing so will trigger this error code.

```
  <ValidationResults>
    <ValidationResultType>B2BERR</ValidationResultType>
    <ValidationResultText>SE-number 12345678 from payload is not valid
</ValidationResultText>
  </ValidationResults>
```

Figure 3-12 - SE-number from payload is not valid example

3.3 XSD Errors

These errors are triggered when the xml does not conform to the XSD for the given declaration. These errors are returned before the declaration reaches DMS. The figure below shows the format of such an error. While the error code might look daunting this error simply means to say that the ExportOffice Identification (element 17 02 001 000) did not conform to the XSD.

```
<Code>Error</Code>
<Message>
  [“cvc-length-valid: Value 'DK00560000000000' with length =
  '16' is not facet-valid with respect to length '8' for type
  '#AnonType_ExportOfficeIdentificationIDType'.”], [“cvc-complex-
type.2.2:
  Element 'ns3:identification' must have no element [children], and the
value must be valid.”]
</Message>
```

Figure 3-13 XSD invalid Error examples

3.4 Synchronous Messages

Synchronous messages are responses that you can experience which will be present on the queue and pulled with the notifications. These messages are not considered notifications and are therefore not wrapped in a <NotificationResult> element. The following list of synchronous messages are not an exhaustive list. Each of the listed messages will have a corresponding XSD, which is available on Github here. The examples presented for each message are not the only triggers that will generate those responses, they are selected examples.

3.4.1 ErrorMessageDTO

This is a validation response on submission of a message. It is triggered by a set of validation errors that are caught before reaching DMS. The example below is received when the submitters differ,but it could also be triggered by an incorrect function code for the given procedure.

```
<ErrorMessageDTO>
    <validationResponse>
        <ValidationResults>
            <ValidationResultType>CWM11007</ValidationResultType>
            <ValidationResultText>Relation error: differing submitters</Valida-
tionResultText>
        </ValidationResults>
    </validationResponse>
    <timestamp/>
</ErrorMessageDTO>
```

Figure 3-14 - ErrorMessageDTO example

3.4.2 GenericErrorDTO

This is a response for HTTP errors; 400, 500, 502 and 503. It will be triggered by many different kinds of errors that result in any of the aforementioned HTTP codes. The first example below is triggered by attempting to log-in with an invalid user. The second example below is triggered by an XSD error. It can also be triggered by using an already existing LRN.

```
<Notification>
<genericErrorDTO>
  <message>LRN : TestDoubletLRN has already been submitted by submitter:
12345678</message>
  <timestamp/>
</genericErrorDTO>
</Notification>
```

```
<genericErrorDTO>
```

```
  <message>TypeCode: 'EXA' is wrong for this action. Only TypeCode 'COR' is al-
lowed for this endpoint</message>

  <timestamp/>
</genericErrorDTO>
```

```
<genericErrorDTO>
  <message>Error while parsing given requested period.</message>
  <timestamp/>
</genericErrorDTO>
```

Figure 3-15 - GenericErrorDTO examples

3.5 ValidationResults

This is a response for submitting **XSD** invalid Exit additional messages such as IE590 or IE507. IE547 and IE583 again does not trigger this response, instead those two will trigger a GenericErrorDTO.

The ValidationResults is in most **other** cases wrapped in a ValidationResponse as can be seen in the example of a ValidationResponse. Therefore, it does not have its own XSD, but can be found in the ValidationResponse XSD. Below is an example of an XSD error in a IE590.

```
<ValidationResults>
    <ValidationResultText>cvc-pattern-valid: Value 'null' is not facet-valid with re-
spect to pattern '([0-1][0-9]|[2][0-4])[A-Z]{2}[A-Z0-9]{13}[0-9]|([2][4-9]|[3-9][0-
9])[A-Z]{2}[A-Z0-9]{12}[A-E][0-9]' for type 'MrnContentType05'.</ValidationResultText>
</ValidationResults>
```

Figure 3-16 ValidationResults example

3.6 Missing notifications

Sometimes the user does not receive the expected notifications. The section below describes a scenario that is not specific to either Import, Export or Transit. Scenarios that may occur for specific domains will be in the DMS Import/Export and DMS Transit System Guides respectively.

3.6.1 No notifications at all

The user should always be able to pull notifications from a given time interval. If the user keeps getting empty notifications for a correct time interval, it might be due to system downtime. If system downtime is not announced on **'Driftsmeddelelser'**, the main system or one of the external systems might be down, and the declaration data cannot properly be received by the system. In this case contact Toldstyrelsens Servicedesk at servicedesk@toldst.dk with information on the declaration(s) with missing notifications.

When the system is up and running again, the declarations should be resubmitted. The LRN(s) can be reused until the declaration(s) are accepted.

3.7 Verifying Functionality

To verify the functionality of the declaration types and additional messages, as well as the ability to request and receive notifications, we recommend that you follow the basic test cases for Import, Export and Transit.

3.7.1 XSDs and test cases

Direct links to all the XSD's used for submission of declarations and for the additional messages are described in the DMS Import/Export and DMS Transit System Guides respectively. All XSD's and test cases can be found on The Tax Administration's GitHub.

Appendix

4

4.1 Interacting with the Internal KRIA Mock

It is possible to send a declaration to control by editing the goods item commodity description. The DMS Import/Export and DMS Transit System Guides will provide details on how to interact with the KRIA mock specifically for their respective domains. Please note that the behavior of the KRIA mock may differ from the behavior of the real KRIA system.

4.2 Codelists used in notifications

Additional codelists can also be found in the DMS Import/Export System Guide under section 7.5 and 7.7.

4.2.1 10182 – StatementTypeCode

List of objects representing the type of additional information as specified in SAD box 44 or linked to other List of objects in the BOM such as Validation Results.

Item Value	Item Description
ABC	Conditions of sale or purchase
ACA	Documentary requirements
ACB	Additional Information
ACD	Amendment Text
AFB	Customs Position Motivation
BLF	Control Explanation
CEX	Customs clearance instructions export
CUS	Special Mentions
OVR01	Additional Information for Alternative Duties Calculation
TRR	Tariff Quota Order Number
ZZZ	Mutually Defined

Table 4-1 - Type of additional information related to Validation Results

4.2.2 10211 - StatementCode

List of objects capturing the reason given in the context of taking a customs position:

	Item Description
1	Simplified declaration not supplemented
2	Release was provided, no coverage for final customs debt surplus
3	Invalidation at trader's request
4	Invalidation by customs due to major discrepancies
5	No exit results received
6	Exit result indicating Goods Stopped
7	Invalidation due to lacking coverage
8	Exit cancelled
9	Release for unsupplemented simplified declaration
10	Release during control
11	Declaration being cancelled, however exit result indicates goods exited or stopped
12	Validation errors after finalized duties
13	Release reconfirmed after amendment

Table 4-2 - Reasons for customs positions in StatementCode

UDVIKLINGS OG FORENKLINGS STYRELSEN

www.ufst.dk
ISBN [000-00-0000-0000-0]

Udviklings- og Forenklings-styrelsen er en del af Skatteforvaltningen