|  |
| --- |
| Project: NMK-NCTSp5  B2G Trader Specifications  EUROPEAN DYNAMICS  **July 2024** |

**HISTORY OF THE DOCUMENT**

| Version | Date | Modifications |
| --- | --- | --- |
| 1.00 | 26/07/2024 | First version |
| 1.01 | 10/10/2024 | Additional exchange data specifications |

**Reference Documents**

| **Ref.** | **Title** | **Reference** | **Version** | **Date** |
| --- | --- | --- | --- | --- |
| [R01] | Trader Specs | NMK-NCTSp5-TRS |  |  |

**Table of Contents**

[1 Introduction 5](#__RefHeading___Toc23261_3542253779)

[1.1 Purpose of the document 5](#__RefHeading___Toc23263_3542253779)

[1.2 Intended Audience 5](#__RefHeading___Toc23265_3542253779)

[2 Overview of the B2G message exchanges 6](#__RefHeading___Toc23267_3542253779)

[2.1 Submission of the messages 6](#__RefHeading___Toc23269_3542253779)

[2.2 Query of the status of a specific message 6](#__RefHeading___Toc23271_3542253779)

[2.3 Query using the MRN or Identification Number 7](#__RefHeading___Toc23273_3542253779)

[2.4 Message Downloading 7](#__RefHeading___Toc23275_3542253779)

[3 B2G web services 8](#__RefHeading___Toc23277_3542253779)

[3.1 Authentication 8](#__RefHeading___Toc23279_3542253779)

[3.2 Request headers 9](#__RefHeading___Toc23281_3542253779)

[3.3 HTTP response codes 9](#__RefHeading___Toc23283_3542253779)

[3.4 SubmitMessage 11](#__RefHeading___Toc23285_3542253779)

[3.4.1 Input 11](#__RefHeading___Toc23287_3542253779)

[3.4.2 Output 12](#__RefHeading___Toc23289_3542253779)

[3.5 GetByCommID 12](#__RefHeading___Toc23291_3542253779)

[3.5.1 Input 13](#__RefHeading___Toc23293_3542253779)

[3.5.2 Output 13](#__RefHeading___Toc23295_3542253779)

[3.6 GetByMRN 14](#__RefHeading___Toc23297_3542253779)

[3.6.1 Input 14](#__RefHeading___Toc23299_3542253779)

[3.6.2 Output 15](#__RefHeading___Toc23301_3542253779)

[3.7 GetByID 15](#__RefHeading___Toc23303_3542253779)

[3.7.1 Input 16](#__RefHeading___Toc23305_3542253779)

[3.7.2 Output 16](#__RefHeading___Toc23307_3542253779)

[3.8 GetByMsgID 17](#__RefHeading___Toc23309_3542253779)

[3.8.1 Input 17](#__RefHeading___Toc23311_3542253779)

[3.8.2 Output 17](#__RefHeading___Toc23313_3542253779)

[4 Management of Electronic signatures 20](#__RefHeading___Toc23315_3542253779)

[4.1 Revocation check 21](#__RefHeading___Toc23317_3542253779)

[5 Use Cases 22](#__RefHeading___Toc23319_3542253779)

[5.1 Example of the standard transit procedure – Core Flow (Happy Path) 22](#__RefHeading___Toc23321_3542253779)

[5.2 Follow-up of messages based on the Trader’s identification number 23](#__RefHeading___Toc23323_3542253779)

**Table of Tables**

[Table 1: SubmitMessage request headers 10](#_Toc172901573)

[Table 2: B2G Service Response HTTP codes 12](#_Toc172901574)

[Table 3: SubmitMessage - Input fields 13](#_Toc172901575)

[Table 4: SubmitMessage - Output fields 13](#_Toc172901576)

[Table 5: GetByCommID - Input fields 14](#_Toc172901577)

[Table 6: GetByCommID - Output fields 15](#_Toc172901578)

[Table 5: GetByMRN - Input fields 16](#_Toc172901579)

[Table 5: GetByID - Input fields 17](#_Toc172901580)

[Table 5: GetByMRN - Input fields 18](#_Toc172901581)

[Table 6: GetByMsgID - Output fields 18](#_Toc172901582)

# Introduction

## Purpose of the document

The purpose of this document is to describe the technical specifications for the B2G message exchanges that **Traders**[[1]](#footnote-2) shall implement for their Business to Government (B2G) interface with the NMK-NCTSp5 IT System and the **Customs Authority** (Customs hereafter).

## Intended Audience

The present document is intended to be read by the following people:

* Trader IT Development Teams

# Overview of the B2G message exchanges

The B2G service will expose the following services:

* Submission of the messages that are sent by the Trader to Customs
* Query of the status of a specific message that was sent by the Trader to Customs
* Query of the status of a transaction with MRN or the Trader’s Identification Number (EORI or TIN)
* Downloading of the messages that are sent by the NMK-NCTSp5 system to the Trader

For all the services, the synchronous message pattern will be used.

## Submission of the messages

The request shall contain the message that the Trader wants to submit to Customs. Only one message per request will be allowed. The supported messages are listed in the Trader Specifications the NMK-NCTSp5 system, in the document with title “Trader Specs - Message Exchanges”. The expected content of each message is described in the Trader Specifications of the NMK-NCTSp5 system, in the document with title “Trader Specs – Message Structures”.

As soon as the B2G service receives the message, it will return to the Trader’s IT system the response with the status of the request.

If there were no technical errors in the processing of the message, a unique communication ID, will be included in the response. That unique ID will have to be used for the status follow-up.

## Query of the status of a specific message

The Trader IT system shall query (ref. 3.5) in regular intervals the B2G service, using the unique communication Identifier that was returned by the message submission service, to retrieve the status of their request. If the request is accepted, the response will contain also the MRN.

## Query using the MRN or Identification Number

The Trader IT system may query the B2G service, using the MRN (Movement Reference Number) (ref. 3.5) of a movement or their identification number (ref. 3.7). The B2G service will check the Trader’s access rights and if the Trader has the necessary access rights, the system will return a list with the status and the message ID for all the messages that are linked to the given MRN or Identification Number (e.g. request for additional documents, control notifications, etc.).

## Message Downloading

The service will allow the Trader IT system to download a message (in XML format) using the message ID. Traders will obtain the message ID from the output of one of the status query services.

# B2G web services

The B2G interface provides a suite of stateless, synchronous REST web services. The web services suite allows external systems to submit, query and retrieve data from the NMK-NCTSp5 IT system.

The functionality of those web services is presented in the following paragraphs.

## Authentication

All services will require HTTP basic authentication (username/password).

Additional metadata fields may be passed to the service in HTTP headers.

## Request headers

All services shall pass the following HTTP Headers, in addition to Authorization headers, when calling this service:

| **Name** | **Value** | **Description** | **Required** |
| --- | --- | --- | --- |
| Content-Type | application/json;charset=utf8 | Indicates the request includes a JSON body with UTF-8 encoding. | Yes |
| TRANSACTION\_ID | ${transactionId}  e.g.  "b053ec66-f1f8-4b95-ab31-07cc96f8014b" | A unique identifier (UUID), set by the calling client, for that request, which is used for ‘idempotency’ purposes, preventing duplicate entries. Also used for logging.  This should be the same as the TRANSACTION\_ID belonging to the overall declaration in the NMK-NCTSp5 IT system for consistent logging and tracing. | Yes, only for the SubmitMessage service |
| TARGET | NCTS | Identifies the target system. It shall be always “NCTS” for the NMK-NCTSp5 IT system. | Yes |

Table *1*: SubmitMessage request headers

## HTTP response codes

The table below presents the HTTP codes that will be returned from the B2G services after the processing of the request:

| **HTTP code** | **JSON Body** | **Description** |
| --- | --- | --- |
| 200 | Yes | The request was processed successfully by the B2G service |
| 400[[2]](#footnote-3) | N/A | Bad request. The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing). |
| 401 | {  *“transactionId”:*  *${TRANSACTION\_ID},*  *“message”: “*  *${Authentication error message}”*  } | Unauthorized. The system failed to authenticate the requesting user with the CDEPS IAM. Field “message” will contain the message (if any) generated by the CDEPS server. |
| 403 | N/A | Forbidden |
| 406 | {  *“transactionId”:*  *${TRANSACTION\_ID},*  *“message”: “Error message #1, Error message #2, ...*  } | Not Acceptable. The incoming request has failed validation in the gateway. Field “message” of the response will contain a concatenation of validation errors separated by commas. |
| 500 | {  *“transactionId”:*  *${TRANSACTION\_ID},*  *“message”:*  *“${ERROR\_MESSAGE}*  } | Internal Server Error. The server has encountered a situation it does not know how to handle.  The Trader shall contact the Customs Helpdesk, informing about the content of the field ${ERROR\_MESSAGE} in the response. |
| 503 | N/A | Service Temporarily Unavailable. The Retry-After HTTP header should, if possible, contain the estimated time before the recovery of the service.  The Trader IT system shall retry automatically the request, only if the Retry-After HTTP header is provided. Otherwise, the Trader shall contact the Customs Helpdesk. |

Table *2*: B2G Service Response HTTP codes

## SubmitMessage

The service will be invoked with an HTTP POST request:

**POST** http://${domain}:${port}/ucc-portal/b2g-service/submitMessage

Note: The URL above is for illustrative purposes only. The exact URL will be provided by the Customs Authority.

### Input

The JSON body requires the data elements laid out below:

| **Name** | **JSON Type** | **Description** | **Required** |
| --- | --- | --- | --- |
| id | String | The identification (EORI or TIN) number of the Trader that is calling the service. The service will reject the request if this number is not the real identification number of the calling client. | Yes |
| messageType | String | The message type, as defined in the “Trader Specs – Message Exchanges” document (e.g. IE015) | Yes |
| messageBody | String | The XML body of the submitted message, base64 encoded.  The XML structure of the declaration/notification shall follow the structure defined in the “Trader Specs – Message Structures” document. | Yes |

Table 3: SubmitMessage - Input fields

### Output

When the request is processed successfully, the output will contain the following information:

| **Element name** | **Type** | **Description** |
| --- | --- | --- |
| service | String | The name of the replying service |
| transaction | String | The transaction ID to which this reply corresponds to |
| message | String | Response message |
| commId | String | The communication ID for the follow-up of the request |

Table 4: SubmitMessage - Output fields

When the returned HTTP code is 200, the JSON body would be like in the example shown below:

{

"service": "SubmitMessage",  
 "transaction": "b053ec66-f1f8-4b95-ab31-07cc96f8014b",

"message": "Message submission OK",

"commId": "a098e03f-37bc-4449-9ffe-35f0f83e5494"

}

}

## GetByCommID

The service will be invoked with an HTTP POST request:

**POST** http://${domain}:${port}/ucc-portal/b2g-service/getByCommId

Note: The URL above is for illustrative purposes only. The exact URL will be provided by the Customs Authority.

### Input

The JSON body of the “getByCommId” service requires the data elements laid out below:

| **Name** | **JSON Type** | **Description** | **Required** |
| --- | --- | --- | --- |
| id | String | The identification (EORI or TIN) number of the Trader that is calling the service. The service will reject the request if this number is not the real identification number of the calling client. | Yes |
| commId | String | The communication ID that was returned from the service SubmitMessage | Yes |

Table 5: GetByCommID - Input fields

### Output

When the request is processed successfully, the output will contain the following information:

| **Element name** | **Type** | **Description** |
| --- | --- | --- |
| service | String | The name of the replying service |
| msgId | String | Unique Message Identifier |
| mrn | String | The MRN of the message (only if it is assigned) |
| msgType | String | The type of the message as provided in the Trader specifications (e.g. IE015) |
| commId | String | The unique communication identifier for this message |
| notification | String | Text message to return if something goes wrong (e.g. Unexpected Error). |

Table 6: GetByCommID - Output fields

**Note:** If an element is empty or not available, it will not be included in the output.

A successful response would be like in the example shown below:

{

"service": "GetByCommID",

"message": {

"msgId": " 01ff01ee-7b5b-40c9-8968-e5f2166adb78",

"mrn": "23IE01000000G7EAR4",

"msgType": "IE015",

"commId": "a098e03f-37bc-4449-9ffe-35f0f83e5494"

}

}

In case of failure, the response structure will be as below:

{

“service”: “GetByCommID”,

“errorMessage”: “*${ERROR\_MESSAGE\_FROM\_SERVER}”*

}

## GetByMRN

The service will be invoked with an HTTP POST request:

**POST** http://${domain}:${port}/ucc-portal/b2g-service/getByMRN

Note: The URL above is for illustrative purposes only. The exact URL will be provided by the Customs Authority.

### Input

The JSON body of the “getByMRN” service requires the data elements laid out below:

| **Name** | **JSON Type** | **Description** | **Required** |
| --- | --- | --- | --- |
| id | String | The identification (EORI or TIN) number of the Trader that is calling the service. The service will reject the request if this number is not the real identification number of the calling client. | Yes |
| mrn | String | The MRN of the requested movement | Yes |

Table 7: GetByMRN - Input fields

### Output

List of messages that are related to the given MRN. For each one of the messages the system will return the available fields defined in Table 6: GetByCommID - Output fields.

An example is shown below:

{

"service": "GetByMRN"

"messages":

[{  
 "msgId": "01ff01ee-7b5b-40c9-8968-e5f2166adb78",

"mrn": "23IE01000000G7EAR4",

"msgType": "IE015",

"commId": "a098e03f-37bc-4449-9ffe-35f0f83e5494"

},

{  
 "msgId": "24071200662255027717",

"msgType": "IE928",

"commId": "f018e03e-99ae-1220-0dde-25f0a62e5495"

}

]

}

In case of failure, the response structure will be as below:

{

“service”: “GetByCommID”,

“errorMessage”: “*${ERROR\_MESSAGE\_FROM\_SERVER}”*

}

## GetByID

The service will be invoked with an HTTP POST request:

**POST** http://${domain}:${port}/ucc-portal/b2g-service/getByID

Note: The URL above is for illustrative purposes only. The exact URL will be provided by the Customs Authority.

### Input

The JSON body of the “getByMRN” service requires the data elements laid out below:

| **Name** | **JSON Type** | **Description** | **Required** |
| --- | --- | --- | --- |
| id | String | The identification (EORI or TIN) number of the Trader that is calling the service. The service will reject the request if this number is not the real identification number of the calling client. | Yes |
| startDate | dateType | The start date (format: yyyy-mm-dd) of the searched period | Yes |
| endDate | dateType | The end date (format: yyyy-mm-dd) of the searched period | Yes |

Table 8: GetByID - Input fields

### Output

List of messages that are related to the identification (EORI or TIN) number of the caller for the given time period. For each message, the output will include the information listed in Table 6: GetByCommID - Output fields.

An example is shown below:

{

"service": "GetByID"

"message": {  
 "msgId": "01ff01ee-7b5b-40c9-8968-e5f2166adb78"

"mrn": "24IE01000000G7EAR4"

"msgType": "IE015"

"commId": "a098e03f-37bc-4449-9ffe-35f0f83e5494"

}

"message": {  
 "msgId": "06f2295d-3073-4929-8406-905e20e8e55"

"msgType": "IE015"

"mrn": "24IE0004000020Z0J5"

"commId": "f018e03e-99ae-1220-0dde-25f0a62e5495"

}

}

In case of failure, the response structure will be as below:

{

“service”: “GetByCommID”,

“errorMessage”: “*${ERROR\_MESSAGE\_FROM\_SERVER}”*

}

## GetByMsgID

The service will be invoked with an HTTP POST request:

**POST** http://${domain}:${port}/ucc-portal/b2g-service/getByMsgId

Note: The URL above is for illustrative purposes only. The exact URL will be provided by the Customs Authority.

### Input

The JSON body of the “getByMsgID” service requires the data elements laid out below:

| **Name** | **JSON Type** | **Description** | **Required** |
| --- | --- | --- | --- |
| id | String | The identification (EORI or TIN) number of the Trader that is calling the service. The service will reject the request if this number is not the real identification number of the calling client. | Yes |
| msgId | String | The message identifier returned from one of the following services: GetByCommID, GetByMRN, GetByID. | Yes |

Table 9: GetByMRN - Input fields

### Output

When the request is processed successfully, the output will contain the following information:

| **Element name** | **Type** | **Description** |
| --- | --- | --- |
| service | String | The name of the replying service |
| msgId | String | Unique Message Identifier |
| mrn | String | The MRN of the message (only if it is assigned) |
| msgType | String | The type of the message as provided in the Trader specifications (e.g. IE015) |
| commId | String | The unique communication identifier for this message |
| messageBody | String | The message in XML format (base64 encoded) |

Table 10: GetByMsgID - Output fields

A successful response would be like in the example shown below:

{

"service": "GetByMsgID",

"message": {

"msgId": " 01ff01ee-7b5b-40c9-8968-e5f2166adb78",

"mrn": "23IE01000000G7EAR4",

"msgType": "IE028",

"commId": "a098e03f-37bc-4449-9ffe-35f0f83e5494",

"msgBody": " PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48bnMyOkNDMDI4QyB4bWxuczpuczI9Imh0dHA6Ly9uY3RzLmRndGF4dWQuZWMiIFBoYXNlSUQ9Ik5DVFM1LjAiPgogICAgPG1lc3NhZ2VTZW5kZXI-TlRBLklFPC9tZXNzYWdlU2VuZGVyPgogICAgPG1lc3NhZ2VSZWNpcGllbnQ-SUUxMzAwTkNUUzwvbWVzc2FnZVJlY2lwaWVudD4KICAgIDxwcmVwYXJhdGlvbkRhdGVBbmRUaW1lPjIwMjQtMDctMjVUMTc6NTg6MTE8L3ByZXBhcmF0aW9uRGF0ZUFuZFRpbWU-CiAgICA8bWVzc2FnZUlkZW50aWZpY2F0aW9uPjRlYzljNWU4ZWZlZjY4PC9tZXNzYWdlSWRlbnRpZmljYXRpb24-CiAgICA8bWVzc2FnZVR5cGU-Q0MwMjhDPC9tZXNzYWdlVHlwZT4KICAgIDxjb3JyZWxhdGlvbklkZW50aWZpZXI-ZjYwZjIzYWJhNTQyMjU8L2NvcnJlbGF0aW9uSWRlbnRpZmllcj4KICAgIDxUcmFuc2l0T3BlcmF0aW9uPgogICAgICAgIDxMUk4-SUUyNDA2MTgwMDAwMjI5OTU1MDAyMTwvTFJOPgogICAgICAgIDxNUk4-MjRJRTAwMDQwMDAwMjBZMks3PC9NUk4-CiAgICAgICAgPGRlY2xhcmF0aW9uQWNjZXB0YW5jZURhdGU-MjAyNC0wNy0yNTwvZGVjbGFyYXRpb25BY2NlcHRhbmNlRGF0ZT4KICAgIDwvVHJhbnNpdE9wZXJhdGlvbj4KICAgIDxDdXN0b21zT2ZmaWNlT2ZEZXBhcnR1cmU-CiAgICAgICAgPHJlZmVyZW5jZU51bWJlcj5JRTAwMDQxMDwvcmVmZXJlbmNlTnVtYmVyPgogICAgPC9DdXN0b21zT2ZmaWNlT2ZEZXBhcnR1cmU-CiAgICA8SG9sZGVyT2ZUaGVUcmFuc2l0UHJvY2VkdXJlPgogICAgICAgIDxpZGVudGlmaWNhdGlvbk51bWJlcj5JRTEzMDBOQ1RTPC9pZGVudGlmaWNhdGlvbk51bWJlcj4KICAgICAgICA8bmFtZT5UcmFkZXIgTkNUUzwvbmFtZT4KICAgICAgICA8QWRkcmVzcz4KICAgICAgICAgICAgPHN0cmVldEFuZE51bWJlcj5NYWluIHN0cmVldDwvc3RyZWV0QW5kTnVtYmVyPgogICAgICAgICAgICA8cG9zdGNvZGU-MTIwMDwvcG9zdGNvZGU-CiAgICAgICAgICAgIDxjaXR5PkR1YmxpbjwvY2l0eT4KICAgICAgICAgICAgPGNvdW50cnk-SUU8L2NvdW50cnk-CiAgICAgICAgPC9BZGRyZXNzPgogICAgPC9Ib2xkZXJPZlRoZVRyYW5zaXRQcm9jZWR1cmU-CjwvbnMyOkNDMDI4Qz4"

}

}

In case of failure, the response structure will be as below:

{

“service”: “GetByCommID”,

“errorMessage”: “*${ERROR\_MESSAGE\_FROM\_SERVER}”*

}

# Management of Electronic signatures

Whenever a new declaration is being submitted by an Economic Operator using the service **submitMessage**, the system will expect the NCTS payload (the declaration content) to be digitally signed in accordance to the provisions of W3C *XML Signature Syntax and Processing Version 1.1*.

In paragraph 3.4.1, Table 3, the *messageBody* part is Base64-encoded content of a B2GMESSAGE structure defined in the B2GMessage.xsd file (part of the Trader Specs).

The B2GMESSAGE is composed of two elements:

* A *Signature* data element, containing the digital signature
* A *any* data element, that is the actual NCTS declaration and that will be digitally signed.

The *Signature* element must contain all the necessary information that will allow the B2G Gateway to validate the signature. After digitally signing the content of the NCTS declaration, the B2GMESSAGE element will have to be Base64 encoded and assigned to the *messageBody* field in the **submitMessage** request.

When the **submitMessage** service is invoked, the Gateway will perform (among others) digital signature / certificate validation according to the steps below:

* **The certificate is signed by KIBS / MK Telecoms**. This validation ensures that the certificate selected by the user is authentic and has in fact been produced by either KIBS or MK Telecoms.
* **The certificate has not been revoked** - This validation is described in detail in the paragraph 4.1
* **The certificate's serial number and the issuer are not associated with another physical person / legal person**. This validation will ensure that the certificate cannot be registered to two different persons at the same time.
* **The certificate is active** - If the certificate is expired or not yet active, the declaration submission will fail.
* **The certificate belongs to the user**.
* **The full name of the subject matches the name of the submitting user** – During the submission of a declaration, this validation is performed to ensure that the submitter’s name (as recorded in the declaration) matches the full name of the user as recorded in the Subject field of the digital certificate. If the names do not match, the submission will fail.
* **VAT number validation** -
  + **For KIBS certificates**, the Company VAT Number is recorded in the subject field of a certificate after the SERIALNUMBER key: CN=John Doe, GIVENNAME=John, SURNAME=Doe, EMAILADDRESS=john.doe@john.doe.com, C=MK, SERIALNUMBER=4030005545553
  + **For MK Telecom certificates**, the VAT Number is recorded in the subject field of a certificate after the “:” symbol: CN=John Doe, GIVENNAME=John, SURNAME=Doe, EMAILADDRESS=john.doe@john.doe.com, C=MK, [SOME TEXT]: 4030005545553

### Revocation check

Revocation verification is performed for all certificates against the CRL services of the corresponding CAs. The Gateway will regularly download and store a copy of the latest versions of the CRLs in the local database. The frequency with which these copies will be updated will be configured as a startup parameter and will have to be agreed upon with the Customs Authority.

# Use Cases

This section presents some typical use cases, implemented using the B2G interface. For the description and the exact name of the XML elements that contain the exchanged information, please refer to the “Trader Specs – Message Structures” document of the Trader Specifications.

## Example of the standard transit procedure – Core Flow (Happy Path)

1. The Holder of the Transit Procedure submits a transit declaration to the Office of Departure with the ‘Declaration Data’ (IE015) message using the “**SubmitMessage**” B2G service.
2. The system returns the commID of the submitted message.
3. The declaration is syntactically valid
4. *The state of the movement at the Office of Departure is set to “None”*
5. The declaration is logically valid
6. The declaration is submitted under normal procedure
7. Trader requests an update of its message using the “**GetbyCommID**” B2G service and the commID that was received in step 2.
8. The system returns the output including the message type, its MRN and its messageID.
9. NCTS acknowledges the reception of the transit declaration with the ‘Positive Acknowledge’ (IE928) message
10. *The state of the movement at the Office of Departure is set to “Submitted”*
11. NCTS communicates the MRN to the Holder of the Transit Procedure with the ‘MRN Allocated’ (IE028) message
12. *The state of the movement at the Office of Departure is set to “Accepted”*
13. *The state of the movement at the Office of Departure is set to “Guarantee under registration”*
14. The guarantee is accepted.
15. *The state of the transit declaration at the Office of Departure is set to “Guarantee registered”*
16. The Office of Departure decide to release the movement
17. *The state of the transit declaration at the Office of Departure is set to “Movement released”*
18. NCTS sends the ‘Release for Transit’ (IE029) message to Holder of the Transit Procedure
19. *The state of the movement at the Office of Departure is set to “*[*Movement written off*](#_Office_of_Departure_1)*”*
20. NCTS sends the ‘Write-Off Notification’ (IE045) message to the Holder of the Transit Procedure.
21. Trader requests an update of its movement using the “**GetbyMRN**” B2G service and the MRN that was received in step 8.
22. The system returns the output including all the messages that had been generated for this movement.

## Follow-up of messages based on the Trader’s identification number

The follow-up of the messages based on the Trader’s identification number correspond to the dashboard view of the user interface.

1. Trader sends a “**GetByID**” B2G service indicating the identification number and the start and end date of the monitoring period.
2. The system returns a list with all the messages that are available for that identification number within the given timeframe. For each message the reply contains the available metadata (commID, messageID, messageType, MRN)
3. The Trader IT system updates its internal state with the received messages, filters the returned messages based on their type and for each one of the messages that needs to be monitored issues a separate “GetByMsgID” B2G service.
4. For each “GetByMsgID” call, the system returns the body of the requested message.
5. The Trader IT system Trader is processing the received messages and updates its internal stat (e.g. if the Trader is a temporary storage facility, the Trader IT system will extract the goods items identifiers and/or Container identifiers of all the accepted temporary storage declarations that bring goods to their facility).

**END OF THE DOCUMENT**

1. Throughout this document the term **Trader** refers to the IT system of the economic operators that will use the B2G (system-to-system) interface for the submission and the follow-up of their declarations. [↑](#footnote-ref-2)
2. Error code 400 shall not occur on a production environment. [↑](#footnote-ref-3)