

AGREED BY

APPROVED BY

_____ / _____
2021

_____ / _____
2021

Orig. Inv. No.	Signature and date	Dupl. Inv. No.	Signature and date
Orig. Inv. No.	Repl. Inv. No.	Dupl. Inv. No.	Signature and date

OMS-CLOUD 3.1
Special software
Programmer's guide

API version 2.0

Revision 19

KZ 15861920.620111-04 33 01

Approval Sheet

2021

APPROVED

KZ 15861920.620111-04 33 01 LU

OMS-CLOUD 3.1

Special software

Programmer's guide

API version 2.0

Revision 19

KZ 15861920.620111-04 33 01

Sheets 133

Orig. Inv. No.	Signature and date	Dupl. Inv. No.	Repl. Inv. No.	Signature and date

2021

ABSTRACT

This document titled “OMS-CLOUD 3.1. Special software. Programmer’s guide” KZ 15861920.620111-04 33 01 is intended to introduce the functions of the application program interface of special software (SSW) of the automated system “OMS-CLOUD 3.1” (OMS Cloud AS 3.1). The document is developed in compliance with GOST 19.504-79 “Unified Software Documentation System. Programmer’s Guide.

CONTENTS

1. PROGRAM FUNCTION.....	6
1.1.Program Function	6
1.2.Software Functionalities	6
2. SOFTWARE APPLICATION	7
2.1.Hardware Used.....	7
2.1.1. Personal Computer Requirements for Operation of Software Package Included in SSW of OMS Cloud 3.1 AS Executed on Hardware of SSW of OMS Cloud 3.1 AS....	7
2.1.2. PC Requirements for Operation of Web-Interface Module Software Package Included in the SSW of OMS Cloud 3.1 AS	8
2.2.Software Required for Software Package Functioning	9
2.2.1. Software Required for Functioning a Software Package included in the SSW of OMS Cloud 3.1 AS Executed on Hardware of OMS Cloud 3.1 AS.....	9
2.2.2. Software Required for Functioning of Web-Interface Module Software Package Included in OMS Cloud 3.1 AS SSW	9
2.3.Infoware Required for Software Package Use.....	10
2.3.1. Generating GS1 DataMatrix.....	10
2.3.2. Requirements for JSON Format Processing	10
3. SOFTWARE FEATURES	11
3.1.Intended Use	11
3.2.Operation Conditions	12
3.3.Software Verification and Self-recovery Means	12
3.3.1. Software Verification Means	12
3.3.2. Self-recovery Means	12
4. PROCEDURE REFERENCE.....	13
4.1.Introduction	13
4.2.Calling Sequence of OMS Methods.....	14
4.3.Description of the MC emission process.....	16
4.3.1. “01.01.00.00 Create marking code emission order”.....	17
4.3.2. “01.02.00.00 Get a status of MC array from the order”	19
4.3.3. “01.03.00.00 Obtain MC from the order”	21
4.3.4. “01.04.00.00 Send report on MC usage”	23
4.3.5. “01.05.00.00 Send Report on MC Aggregation”	25
4.4.OMS API extensions	27
4.4.1. Method “Create marking code emission order”	28
4.4.1.1. Request.....	28
4.4.1.2. Request Response.....	43
4.4.2. Method <Send a report on MC dropout/rejection>.....	44
4.4.2.1. Request.....	44
4.4.2.2. Request Response.....	48
4.4.3. Method <Send Report on MC Aggregation>.....	49
4.4.3.1. Request.....	49
4.4.3.2. Request Response.....	55
4.4.4. Method <Send Report on MC Utilization (Application)>.....	56
4.4.4.1. Request.....	56
4.4.4.2. Request Response.....	61
4.4.5. Method <Close suborder/order>.....	62
4.4.5.1. Request.....	62
4.4.5.2. Request Response.....	64
4.4.6. Method <Obtain MC from the order>.....	65
4.4.6.1. Request.....	65

4.4.6.2. Request Response.....	67
4.4.7. Method «Get a status of MC array from the order»	68
4.4.7.1. Request.....	68
4.4.7.2. Request Response.....	69
4.4.8. Method «Get Orders Status»	71
4.4.8.1. Request.....	71
4.4.8.2. Request Response.....	72
4.4.9. Method «Receive aggregation information»	74
4.4.9.1. Request.....	74
4.4.9.2. Request Response.....	75
4.4.10. Method «Obtain the Report Processing Status»	79
4.4.10.1. Request.....	79
4.4.10.2. Request Response.....	80
4.4.11. Method «Send APCS log files».....	81
4.4.11.1. Request.....	81
4.4.11.2. Request Response.....	82
4.4.12. Method «Check OMS availability».....	83
4.4.12.1. Request.....	83
4.4.12.2. Request Response.....	84
4.4.13. Method «Get security marker by username and password»	85
4.4.13.1. Request.....	85
4.4.13.2. Request Response.....	86
4.4.14. Method “Obtain OMS and API version”	87
4.4.14.1. Request.....	87
4.4.14.2. Request Response.....	87
4.4.15. Method “Obtain the list of marking code package identifiers”	88
4.4.15.1. Restrictions	88
4.4.15.2. Request.....	88
4.4.15.3. Request Response.....	89
4.4.16. Method «Re-obtain marking codes from marking code order»	91
4.4.16.1. Restrictions	91
4.4.16.2. Request.....	91
4.4.16.3. Request Response.....	93
4.4.17. Method “Receive receipt by unique document identifier”	94
4.4.17.1. Request.....	94
4.4.17.2. Response.....	95
5. INPUT AND OUTPUT DATA.....	97
5.1.Character, organization and preliminary preparation of input and output data.....	97
5.1.1. Data sources.....	97
5.1.2. Methods for arranging collection, transmission, monitoring and correction of the information	97
5.2.Format, Description and Method of Encoding Input and Output Data When Using API	99
5.3.Catalogs Available Through API.....	100
5.3.1. Catalogs to operate with the marking codes	100
5.3.1.1. Catalog «Method of Goods Release into Circulation»	100
5.3.1.2. Catalog «Method of Individual Serial Number Generation».....	100
5.3.1.3. Catalog «Manufacturing method»	100
5.3.1.4. Catalog «MC Templates»	101
5.3.1.5. Catalog «MC Array Status».....	103
5.3.1.6. Catalog «Aggregation Type»	104
5.3.1.7. Catalog «MC Buffer Status»	105

5.3.1.8. Catalog «Report Processing Status».....	106
5.3.1.9. Catalog «Type of Utilization».....	107
5.3.1.10. Catalog «Order status»	108
5.3.1.11. Catalog «Disposal Reason».....	109
5.3.1.12. Catalog «Marking Code Type»	110
5.3.1.13. Catalog «Goods Groups»	110
5.3.1.14. Catalog «International Classifier of Countries».....	111
6. MESSAGES	118
6.1.Messages to Operator via GUI.....	118
6.1.1. Information windows	118
6.2.Error Format and Codes.....	119
6.2.1. Error format	119
6.2.2. Description of Errors	120
APPENDIX	121
LIST OF TERMS	123
LIST OF ABBREVIATIONS	125
LIST OF PICTURES.....	126
LIST OF TABLES.....	128

1. PROGRAM FUNCTION

1.1. Program Function

The specialized software is intended to enable the performance of target functionalities of OMS-Cloud 3.1 AS including:

- 1) Performing orders for marking code emission.
- 2) Provision of emitted marking codes to apply to the products.
- 3) Verification of MC application.
- 4) Aggregation of ready and packed products marked with identification means (IM).
- 5) Disposal of defective goods.

1.2. Software Functionalities

SSW of OMS Cloud 3.1 AS ensures performance of the following functionalities:

- 1) Receiving of the orders and provision of the emitted marking codes.
- 2) Receiving of marking code use (application) reports and registration of marking code status in the marking system.
- 3) Receiving of marking code aggregation reports and registration of packaged products aggregation in the marking system.
- 4) Receiving of products rejection reports and registration of rejected products in the marking system.

2. SOFTWARE APPLICATION

2.1. Hardware Used

2.1.1. Personal Computer Requirements for Operation of Software Package Included in SSW of OMS Cloud 3.1 AS Executed on Hardware of SSW of OMS Cloud 3.1 AS

For operation of a software package included in the SSW of OMS Cloud 3.1 AS Executed on Hardware of SSW of OMS Cloud 3.1 AS, the following computer equipment shall be used:

- 1) CPU: Intel x86, 3 GHz;
- 2) Platform: x64-based;
- 3) Main memory capacity: 16 Gb;
- 4) HDD: 100GB;
- 5) Network interface: Ethernet 100 Mbit/s.

Minimum network interface requirements for the OMS:

- 1) 1 static IP address (IP address);
- 2) Subnet mask (netmask);
- 3) Default gateway;
- 4) DNS settings.

Access to OMS server:

- 1) An SSH port (22 by default or any other port configured by the system administrator) shall be open in firewall, as well as ports for the Imagenarium Software web console and the OMS Software Web-interface.

To ensure fail-safe operation, it is recommended to use three physical or virtual machines combined in a cluster. The all machines shall have identical configuration. All machines shall have the same SSH ports.

2.1.2. PC Requirements for Operation of Web-Interface Module Software Package Included in the SSW of OMS Cloud 3.1 AS

To operate the information services provided by the SSW of OMS Cloud 3.1 AS, the following computer equipment shall be used:

1) Single-user computer equipment:

- the AWS of a user with the functional role of “Operator / Administrator of the OMS” (AWS of Operator / Administrator of the OMS) that is a PC with characteristics relevant to the recommended requirements of Microsoft Windows OS 8.1 and later editions, and a network adapter providing infocommunication channel with HCS, OMS-Cloud 3.1 AS.

2.2. Software Required for Software Package Functioning

2.2.1. Software Required for Functioning a Software Package included in the SSW of OMS Cloud 3.1 AS Executed on Hardware of OMS Cloud 3.1 AS

For the system operation, the following basic programs and components shall be installed on the server / virtual machine:

- 1) Linux CentOS 7 operating system;
- 2) SSH Server (authentication by name and password);
- 3) Command line and system program utility packages: (bash, ifconfig, sysctl, curl, yum, systemctl, yum-config-manager, unzip).

For the OMS Cloud 3.1 AS SSW installation, it is necessary to install the Linux CentOS 7 operating system, configure the SSH Server, and utility packages of command line and system programs listed above.

2.2.2. Software Required for Functioning of Web-Interface Module Software Package Included in OMS Cloud 3.1 AS SSW

To operate the information services provided by “Web-Interface Module” software package, part of OMS-Cloud 3.1 AS SSW, the following general software components are required:

- 1) Microsoft Windows OS 8.1 or higher;
- 2) Web browser:
 - Mozilla Firefox 40 or higher.
 - Internet Explorer 9 or higher.
 - Google Chrome 37 or higher.

2.3. Infoware Required for Software Package Use

2.3.1. Generating GS1 DataMatrix

According to GS1 General Specification requirements, before converting to DataMatrix, to properly form GS1 DataMatrix, it is necessary to add ASCII232 symbology attribute to the beginning of the received line of the marking code, otherwise, the technical means will not recognize the code correctly and will not be able to process it. Links to specifications are given below:

1) GS1	General	Specification
(https://www.gs1.org/docs/barcodes/GS1_General_Specifications.pdf);		
2) GS1	DataMatrix	Guideline
(https://www.gs1.org/docs/barcodes/GS1_DataMatrix_Guideline.pdf).		

2.3.2. Requirements for JSON Format Processing

As the marking code contains special symbols, the circulation participants, when integrating their solutions, shall correctly process JSON format with tools compliant with RFC 8259, and never process it as Plain Text. The use of tools compliant with RFC 8259 within such solutions ensures that marking codes with special symbols are transferred and obtained correctly (special symbols are screened).

Likewise, if circulation participants use XML format in their solutions, the special symbols contained in a string shall be converted in accordance with XML specifications.

3. SOFTWARE FEATURES

3.1. Intended Use

OMS-Cloud 3.1 AS SSW is intended to enable the performance of the targeted functionalities of OMS-Cloud 3.1 AS in accordance with the following indicators of the system purpose:

- 1) For the list of the purpose indicators to which the OMS 3.1 must conform, see Table 1.

Table 1 — List of the purpose indicators to which the system must conform

Purpose indicators to which the OMS must conform	Indicators' value
Number of goods items per marking code order	Maximum 10
Number of the marking codes of a goods item (goods code, GTIN) in the marking codes order	Maximum 150,000 marking codes
Number of the marking codes in the marking code application report	Maximum 30,000 marking codes
Number of the marking codes in the MC dropout/rejection report	Maximum 30,000 marking codes

Note: For goods group category “Medicines for human use”, the quantity of goods items per one marking code order shall not exceed one (1 marking codes order - 1 GTIN).

3.2. Operation Conditions

Operation conditions of OMS-Cloud 3.1 AS SSW is round-the-clock throughout the year (24/7/365).

3.3. Software Verification and Self-recovery Means

3.3.1. Software Verification Means

Verification of OMS-Cloud 3.1 AS SSW is performed by means of:

- 1) Built-in diagnostic tools.
- 2) Relevant software tools validation.

3.3.2. Self-recovery Means

OMS-Cloud 3.1 AS SSW self-recovery is performed by means of:

- 1) Built-in self-recovery tools.
- 2) Automatic back-up tools.
- 3) Built-in operating system tools.

4. PROCEDURE REFERENCE

4.1. Introduction

API REST controller authenticates clients using the so-called client token sent by client in HTTP-request header. The security marker (ClientToken) is sent in HTTP header of the “clientToken”.

Some API methods use HTTP POST method for sending data. In this case, use the indication of an additional HTTP header – “Content-Type: application/json”.

OMS API methods use “omsId” identifier of OMS, which is available in OMS settings.

Valid MC symbols are given in Table 2. These symbols shall be used in the following groups of the marking codes data: “Serial Number”, “Key Identifier”, “Verification Code”.

Table 2 – Valid MC Symbols

Valid MC symbols
ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!"%&()'*+,-./_::=<>?

4.2. Calling Sequence of OMS Methods

This section describes OMS API, with interaction via HTTP using JSON format.

The sequence of OMS method call during the creation of a new MC emission order is as follows:

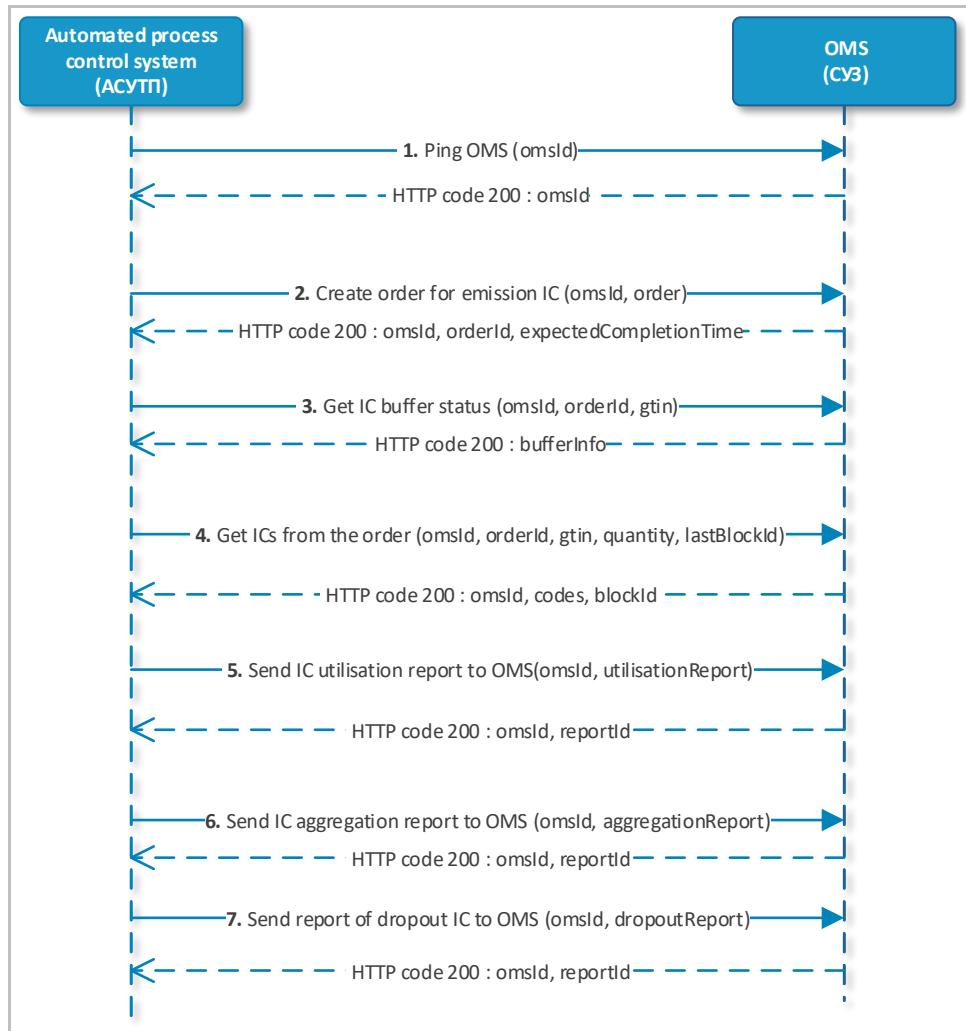
- 1) Check availability of OMS (see section 4.4.12).
- 2) Generate an order for marking code emission (see section 4.4.1).
- 3) Get a status of MC array from the order (see section 4.4.7).
- 4) Obtain MC from the order (see section 4.4.6).
- 5) Send report on MC usage (see section 4.4.4).
- 6) Send report on MC aggregation (see section 4.4.3).
- 7) Send report on MC dropout/rejection (see section 4.4.2).

Diagram of the OMS methods calling sequence is shown in Figure 1.

OMS API also provides auxiliary methods:

- 1) Obtain the report processing status (see section 4.4.10).
- 2) Send APICS log files (see section 4.4.11).
- 3) Obtain order status (see section 4.4.8).
- 4) Get security marker by username and password (see section 4.4.13).
- 5) Close the suborder by the GTIN specified (see section 4.4.5).
- 6) Receive aggregation information (see section 4.4.9).
- 7) Obtain OMS and API version (see section 4.4.14).
- 8) Obtain the list of marking code package identifiers (see section 4.4.15).
- 9) Re-obtain marking codes from marking code order (see section 4.4.16).

Note: In case there are any unused marking codes when the order is closed, a report on the marking code cancellation will be generated and sent, unused MCs will be assigned <ELIMINATED> (unused) status.



Calling Sequence of OMS Methods

Figure 1

4.3. Description of the MC emission process

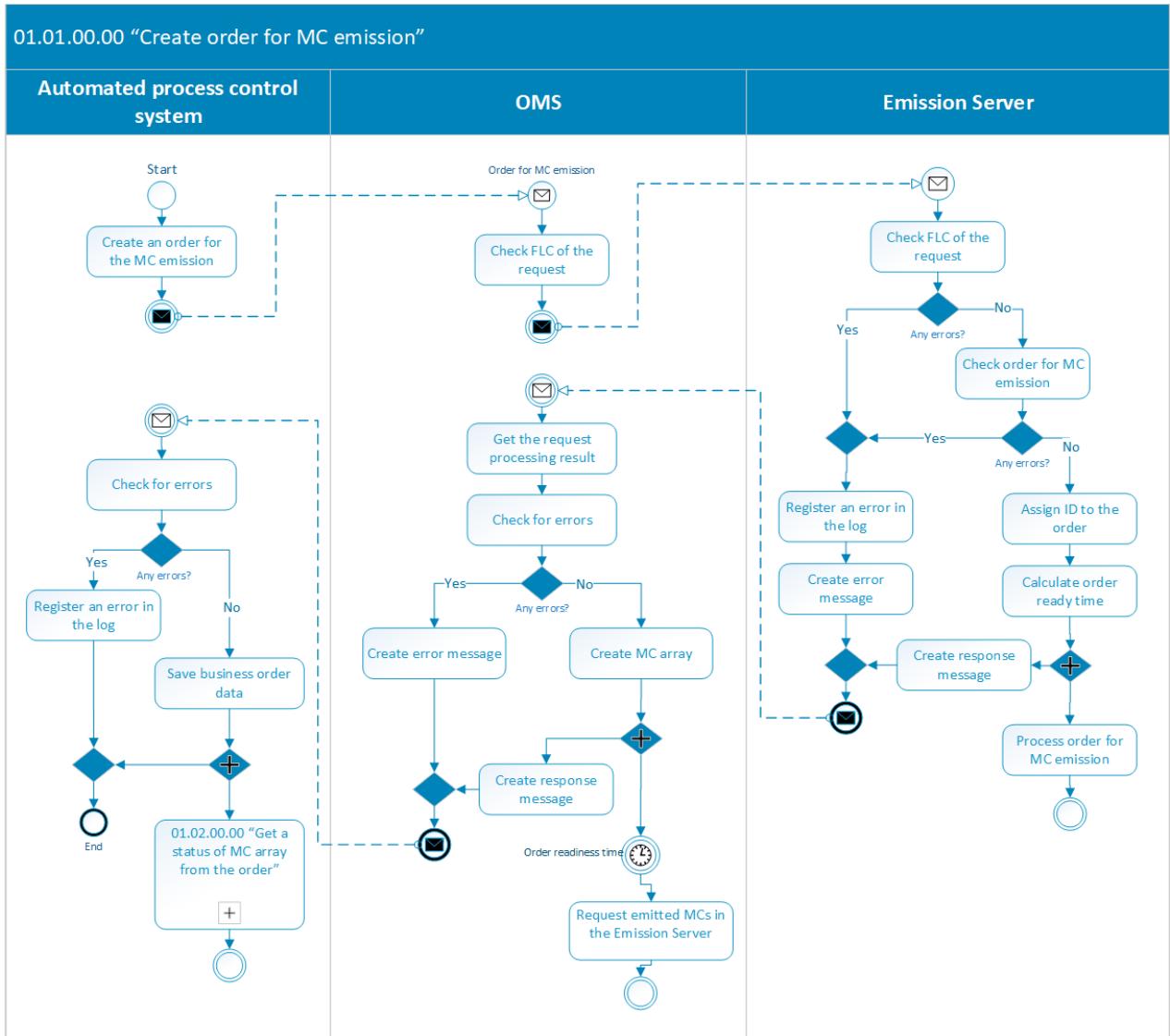
This subsection describes the marking codes emission process. General MC emission process consists of four key stages:

- 1) "01.01.00.00 Create marking code emission order";
- 2) "01.02.00.00 Get a status of MC array from the order";
- 3) "01.03.00.00 Obtain MC from the order";
- 4) "01.04.00.00 Send report on MC usage".

Sending of the aggregation reports and dropout/rejection reports is carried out in the same manner as in "01.04.00.00 Send report on MC usage" (see section 4.3.5).

4.3.1. “01.01.00.00 Create marking code emission order”

For diagram of the MC emission order creation, see Figure 2.



Process “Create marking code emission order”

Figure 2

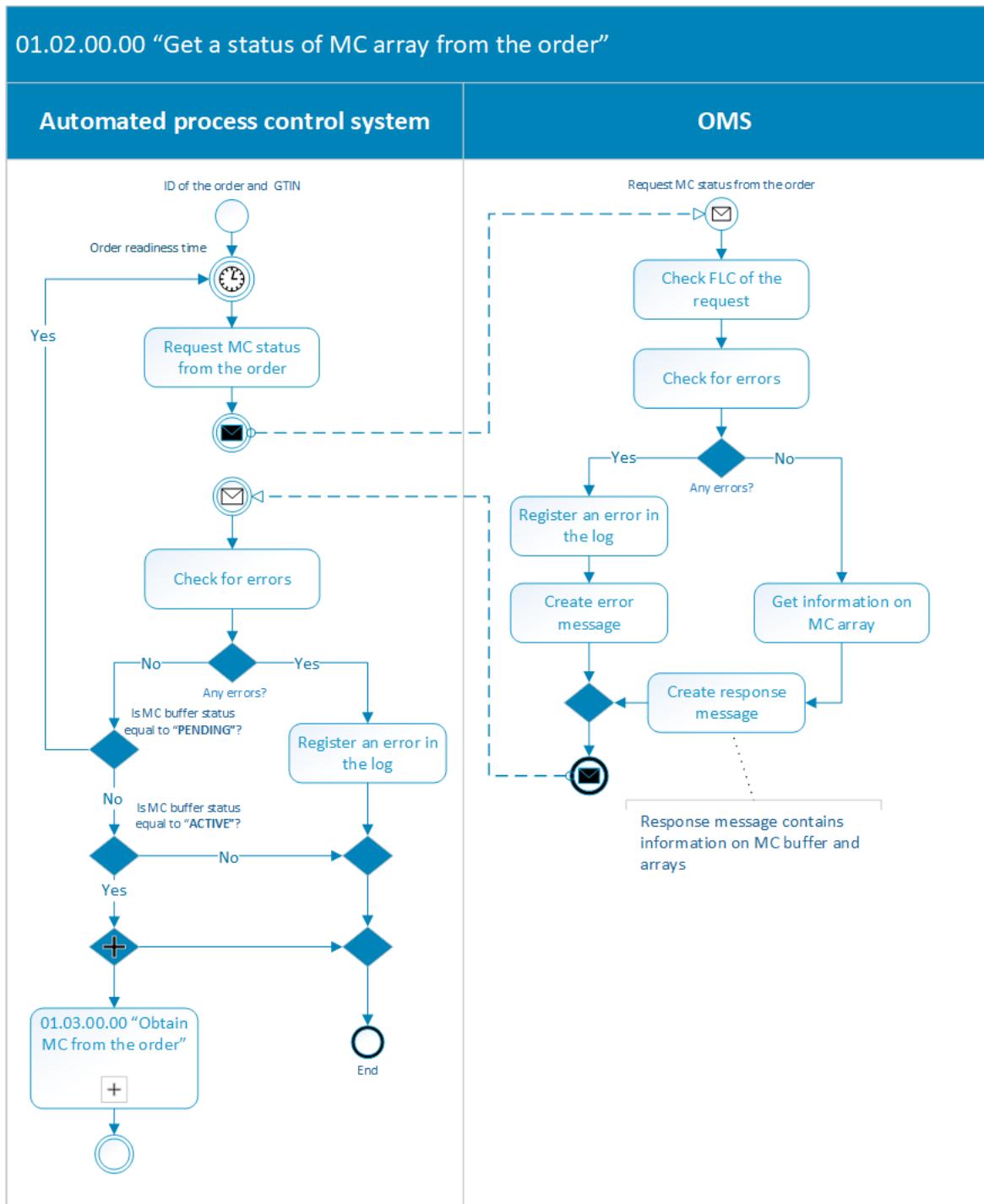
Description:

- 1) APCS generates an order and sends it to the OMS;
- 2) The OMS verifies the request and sends the order to Emission Server;
- 3) Having received the request containing the MC emission order, the Emission Server verifies the request.
 - If request contains errors, the Emission Server will register the error in the log;
 - The Emission Server generates an error message and sends it to the OMS;
 - Then you will be taken to step 7) of the main scenario.
- 4) If there are no errors, the Emission Server will check the MC emission order.

- If request contains errors, the Emission Server will register the error in the log;
 - The Emission Server generates an error message and sends it to the OMS;
 - Then you will be taken to step 7) of the main scenario.
- 5) If there are no errors in the MC emission order, the Emission Server assigns an identifier to the order and calculates the order readiness time;
- The Emission Server sends the order for processing (the action is performed asynchronously);
- 6) The Emission Server generates a response message and sends it to the OMS.
- 7) The OMS receives the request processing result from the Emission Server;
- 8) The OMS checks for errors;
- If there are errors, the OMS will register them in the log;
 - The OMS generates an error message and sends it to APSCS;
 - Then you will be taken to step 11) of the main scenario.
- 9) If there are no errors, the OMS creates the MC array;
- The OMS waits for order readiness and requests MCs emitted in the Emission Server (the action is performed asynchronously).
- 10) The OMS generates a response message and sends to APSCS;
- 11) APSCS receives the request processing result from the OMS;
- 12) APSCS checks for errors;
- If there are errors, APSCS will register them in the log;
 - The process ends.
- 13) If there are no errors, APSCS stores the order data;
- APSCS initiates the execution of 01.02.00.00 “Get a status of MC array from the order” (the action is performed asynchronously).
- 14) The process ends.

4.3.2. “01.02.00.00 Get a status of MC array from the order”

Diagram of process for getting a MC array status can be found in Figure 3.



Process “Get a status of MC array from the order”

Figure 3

Description:

- 1) APCS waits for order readiness;
- 2) APCS forms a request to receive the MC array status and sends it to the OMS;
- 3) OMS verifies the request;

4) The OMS checks for errors:

- If there are errors in the request, the OMS will register the error in the log;
- The OMS generates an error message and sends it to APSCS;
- Then you will be taken to step 7 of the main scenario.

5) The OMS receives information on the MC array;

6) The OMS generates a response message and sends it to APCS;

7) APSCS receives the response message;

8) APSCS checks for errors:

- If there are errors in the request, APSCS will register the error in the log;
- The process ends.

9) If there are no errors, APSCS checks if the MC buffer status equals to “PENDING”:

- If the MC buffer status equals to “PENDING”, APSCS initiates a repeat request for the status of the MC array.
- Then you will be taken to step 1 of the main scenario.

10) If there are no errors, APSCS checks if the MC buffer status equals to “ACTIVE”:

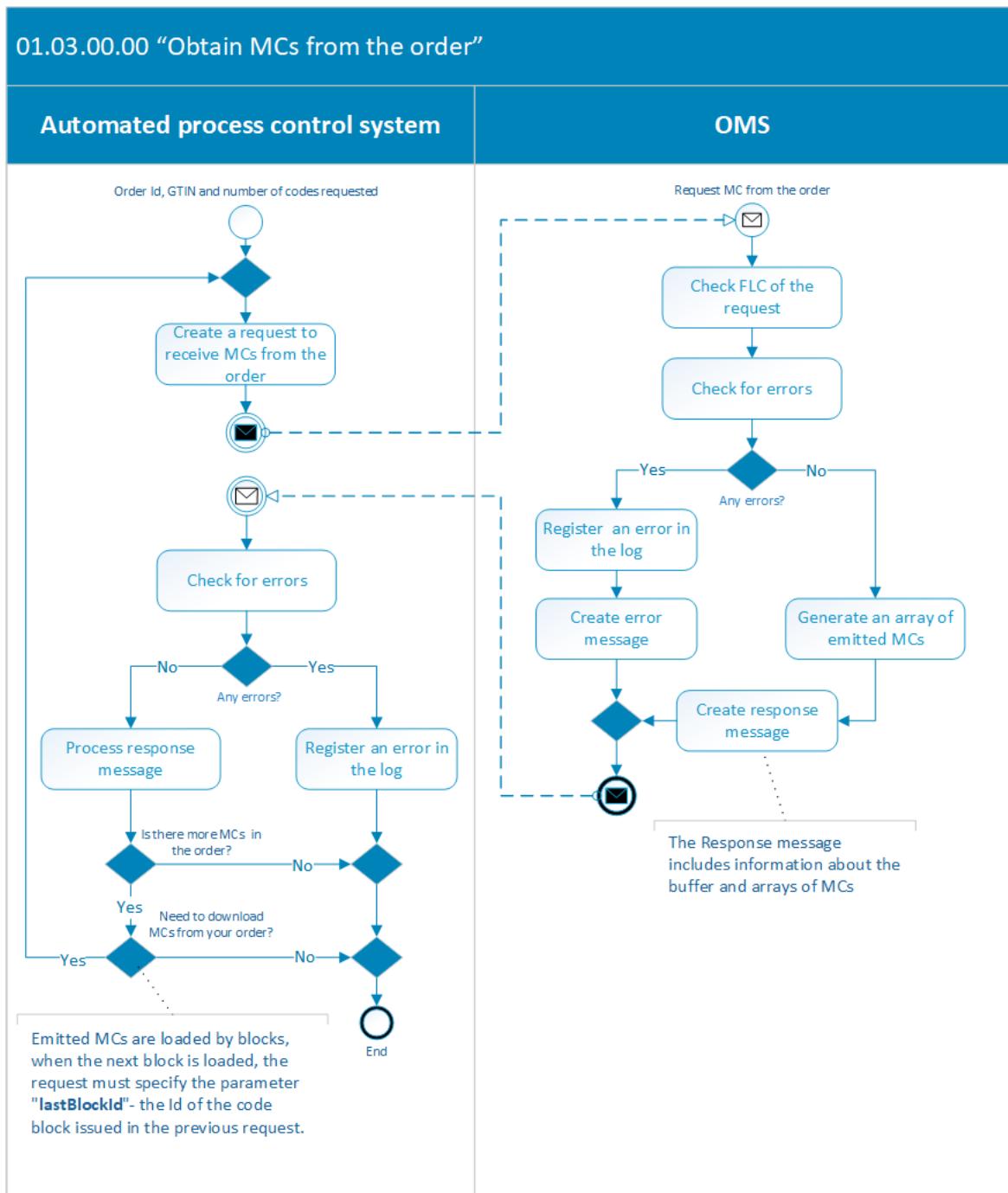
- If the MC buffer status is not equal to “ACTIVE”, the process ends.

11) If the MC buffer status is “ACTIVE”, APSCS initiates the execution of 01.03.00.00 “Obtain MC from the order” (the action is performed asynchronously);

12) The process ends.

4.3.3. “01.03.00.00 Obtain MC from the order”

Diagram of process for getting the MCs from the marking code emission order can be found in Figure 4.



Process “Obtain MC from the order”

Figure 4

Description:

- 1) APCS generates a request to obtain the MCs from the order and sends it to the OMS;
- 2) OMS verifies the request;

3) The OMS checks for errors:

- If there are errors in the request, the OMS will register the error in the log;
- The OMS generates an error message and sends it to APSCS;
- Then you will be taken to step 6 of the main scenario.

4) OMS forms an array of emitted MCs;

5) The OMS generates a response message and sends it to APCS;

6) APSCS receives the response message;

7) APSCS checks for errors:

- If there are errors in the request, APSCS will register the error in the log;
- The process ends.

8) APSCS processes the received message.

9) APSCS checks if there are any other MCs in the order.

- If there are no MCs in the order, the process ends.

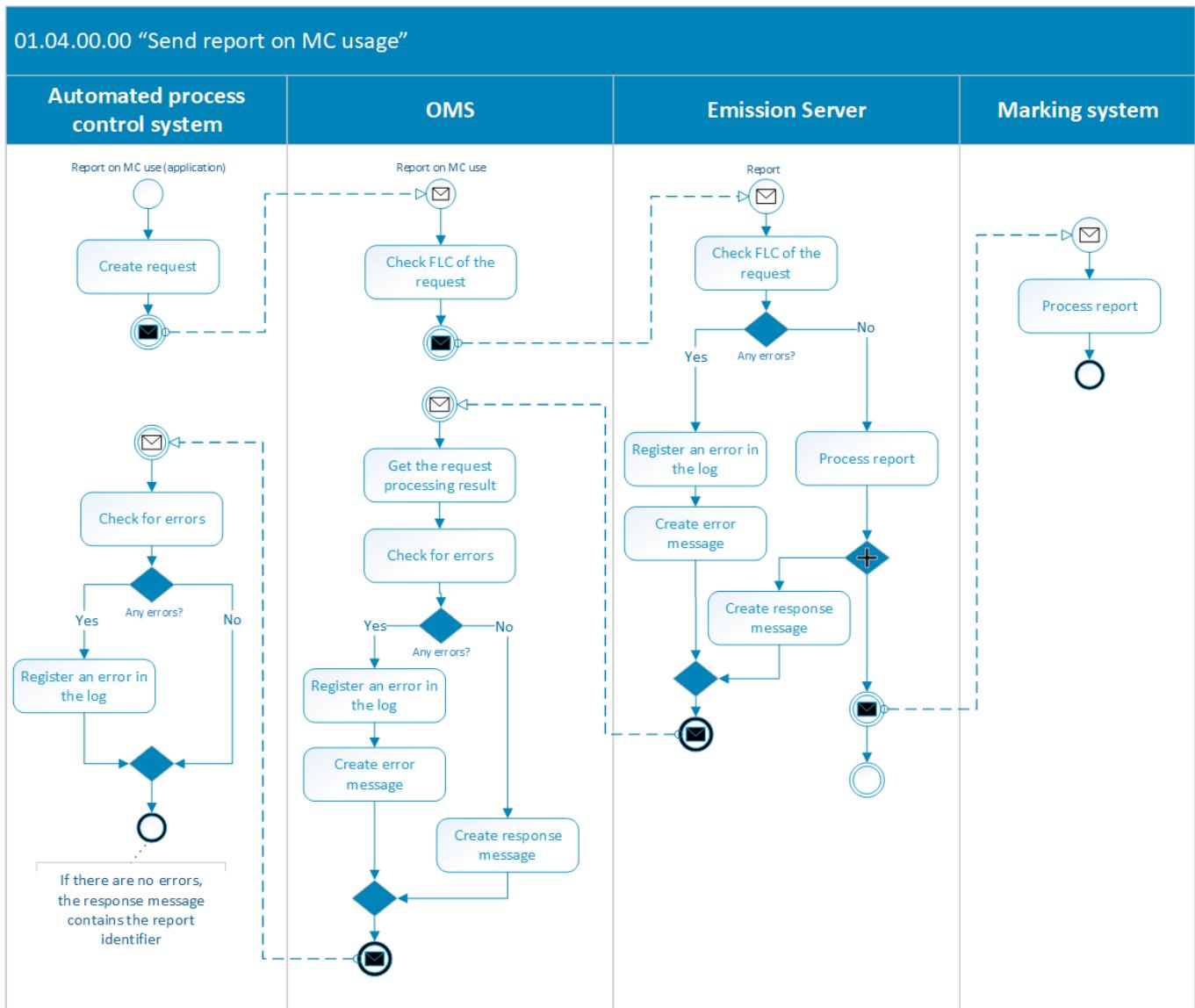
10) If there are MCs in the order, APSCS checks whether loading of remaining MCs is required.

- If it is necessary to load the remaining MCs in the order, APSCS initiates the re-execution of 01.03.00.00 “Obtain MC from the order”;
- Then you will be taken to step 1 of the main scenario.

11) If there is no need to load the remaining MCs in the order, the process ends.

4.3.4. “01.04.00.00 Send report on MC usage”

Diagram of sending the report on MC use can be found in Figure 5.



Process “Send report on MC usage”

Figure 5

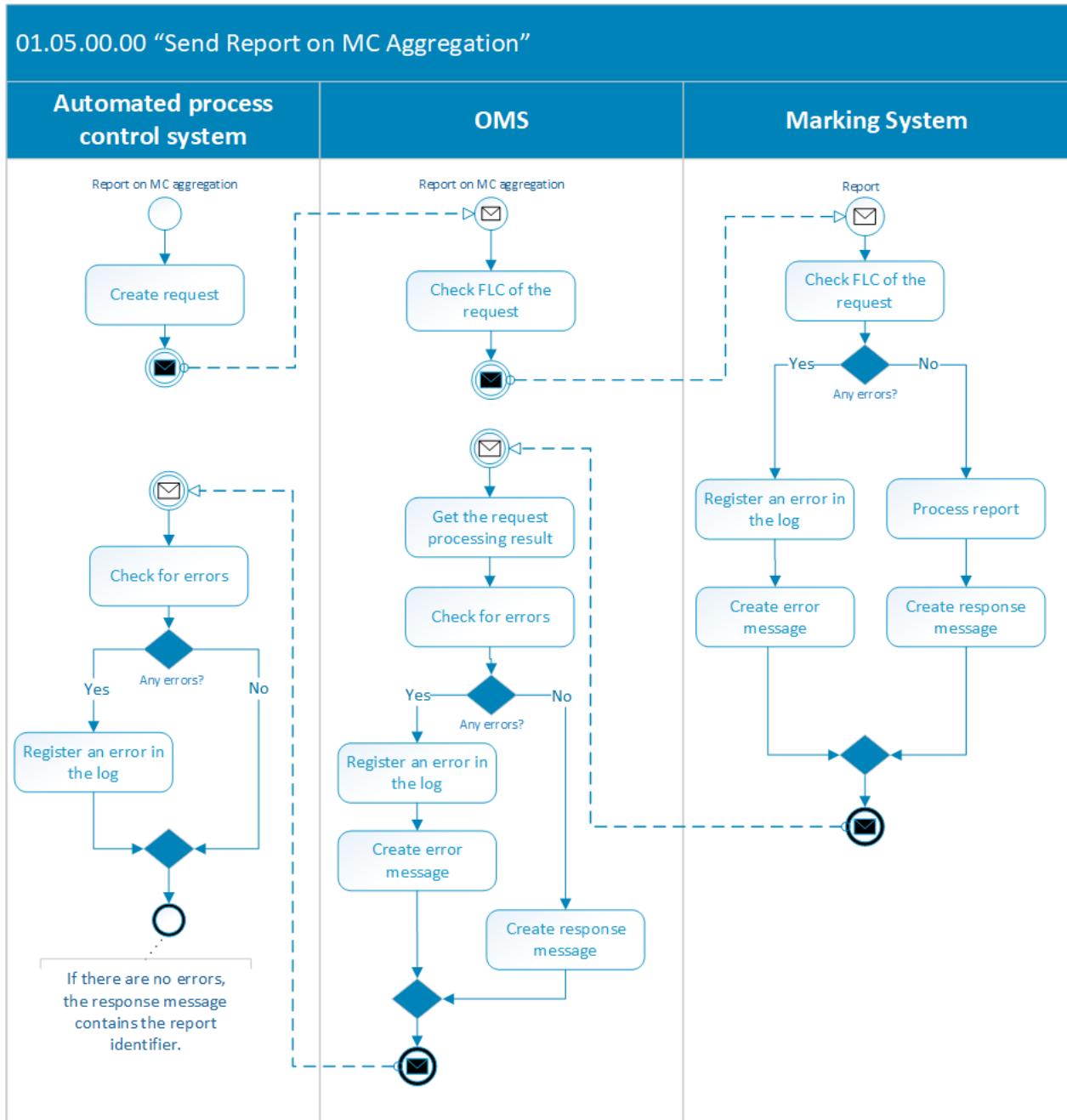
Description:

- 1) APCS generates a request with the report on MC usage and sends it to the OMS;
- 2) The OMS verifies the request and sends a request with the report on MC usage to Emission Server;
- 3) Having received the request with the report on MC usage, the Emission Server verifies the request.
 - If there are errors in the request, the Emission Server will register the error in the log;
 - The Emission Server generates an error message and sends it to the OMS;
 - Then you will be taken to step 6) of the main scenario.

- 4) If there are no errors, the Emission Server processes the report;
 - The Emission Server sends the report for processing to the marking system (the action is performed asynchronously);
- 5) The Emission Server generates a response message and sends it to the OMS;
- 6) The OMS receives the request processing result from the Emission Server;
- 7) The OMS checks for errors;
 - If there are errors, the OMS will register them in the log;
 - The OMS generates an error message and sends it to APSCS;
 - Then you will be taken to step 9) of the main scenario.
- 8) If there are no errors, the OMS generates a response message and sends it to APSCS;
- 9) APSCS receives the request processing result from the OMS;
- 10) APSCS checks for errors;
 - If there are errors, APSCS will register them in the log;
 - The process ends.
- 11) If there are no errors, APSCS saves the identifier of the report, the process ends.

4.3.5. “01.05.00.00 Send Report on MC Aggregation”

Diagram of sending the reports on MC aggregation can be found in Figure 6.



Process “Send report on MC aggregation”

Figure 6

Description:

- 1) APCS generates a request with the report on MC aggregation and sends it to the OMS;
- 2) The OMS verifies the request and sends a request with the report on MC aggregation to the marking system;

- 3) Having received the request with the report on MC aggregation, the marking system verifies the request.
 - If there are errors in the request, the marking system will register the error in the log;
 - The marking system generates an error message and sends it to the OMS;
 - Then you will be taken to step 6) of the main scenario.
- 4) If there are no errors, the marking system processes the report;
- 5) The marking system generates a response message and sends it to the OMS;
- 6) The OMS receives the request processing result from the marking system;
- 7) The OMS checks for errors;
 - If there are errors, the OMS will register them in the log;
 - The OMS generates an error message and sends it to APSCS;
 - Then you will be taken to step 9) of the main scenario.
- 8) If there are no errors, the OMS generates a response message and sends it to APSCS;
- 9) APSCS receives the request processing result from the OMS;
- 10) APSCS checks for errors;
 - If there are errors, APSCS will register them in the log;
 - The process ends.
- 11) If there are no errors, APSCS saves the identifier of the report, the process ends.

4.4. OMS API extensions

OMS API supports extensions for alcohol, tobacco, pharmaceutical industry, and «Footwear», «Clothing items, bed, table, bath and kitchen linens», «Packaged water and sugar-containing beverages» goods groups. Access to the OMS API extensions is provided via URL.

URL structure of the OMS API has the following parameters:

`http://<server-name>[:server-port]/api/v2/{extension}/`

has the following parameters:

- 1) `server-name` — server name or IP address;
- 2) `server-port` — connection port;
- 3) `extension` — URL-parameter defining access to OMS API extensions.

The parameter URL `{extension}` that defines access to extensions of the goods groups, has the following default values:

- 1) `shoes` — parameter URL `{extension}` for the «Footwear» goods group;
- 2) `tobacco` — parameter URL `{extension}` for tobacco industry;
- 3) `alcohol` — parameter URL `{extension}` for alcohol industry;
- 4) `pharma` — parameter URL `{extension}` for pharmaceutical industry;
- 5) `milk` — parameter URL `{extension}` for dairy industry;
- 6) `lp` — parameter URL `{extension}` for light industry.
- 7) `water` — parameter URL `{extension}` for water industry.

4.4.1. Method “Create marking code emission order”

This method is used to create and send the order for MC emission. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP-header with the name “clientToken”.

Notes:

1. One goods item (goods code, GTIN) within one order shall not exceed 150,000 marking codes, and the quantity of goods items within one order shall not exceed 10 (1 order - 10 GTIN).

For pharmaceutical industry, the quantity of goods items per one order shall not exceed 1 (1 order – 1 GTIN).

2. Maximum 100 orders may be active simultaneously. Active orders are the orders with status READY, where at least one suborder (MC buffer) has status ACTIVE, PENDING or EXHAUSTED.

More than 100 orders may not be waiting in the queue as well. Such orders include the orders with status CREATED, PENDING, APPROVED.

If one of the limits is reached, it will not be possible to create the order.

3. In line with note 2, the method can be addressed from one source not more than 100 times per second.

4.4.1.1. Request

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/orders?omsId={omsId}`

Method: POST

Content-type: application/json

clientToken: {clientToken}

Request string parameters are given in Table 3.

Table 3 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes

For the description of JSON format of the request for creation and sending of order for the MC emission (object <Order>), see Table 4.

Table 4 – Description of JSON format of the request for creation and sending of the order for the MC emission, object <Order>

Field	Description	Type	Mandatory
products	List of goods	JSON Array of OrderProduct (Table 5)	Yes
serviceProviderId	Service provider identifier	String(36) UUID	No

Description of the <OrderProduct> object format is given in Table 5.

Table 5 – Format of <OrderProduct> Object

Field	Description	Type	Mandatory
gtin	Goods code (GTIN) of the product	String (14) [0-9]{14}	Yes
quantity	MC Quantity	Integer (\$int32)	Yes
serialNumberType	Serial number generation method	String (See section 5.3.1.2)	Yes
serialNumbers	Array of serial numbers. This field is specified if value "serialNumber = SELF_MADE" (see section 5.3.1.2).	JSON Array of String*	No Conditionally mandatory
templateId	MC template identifier	Integer (\$int32) (see section 5.3.1.4)	Yes
stickerId	Label identifier	String	No Conditionally mandatory

Notes:

1. For <Tobacco products> goods group, the initially set generation scheme and MC-template structure for the specific goods type (GTIN) defined by attribute <serialNumberType> cannot be changed later.
2. "stickerId" field shall be filled in when creating order as part of distribution process.
3. For <Dairy products> goods group, when selecting the value "serialNumber = SELF_MADE", the serial number length shall consist of 5 characters. When marking codes are emitted, the serial number will consist of 6 characters including the country code. The country code is indicated by the Emission Server and shall be entered before the received serial number.

4. For «Clothing items, bed, table, bath and kitchen linens» goods group, when selecting the value “serialNumber = SELF_MADE”, the serial number length shall consist of 12 characters. When marking codes are emitted, the serial number will consist of 13 characters including the country code. The country code is indicated by the Emission Server and shall be entered before the received serial number.

4.4.1.1.1 Extensions for tobacco industry

Description of «Order» object extension for tobacco industry can be found in Table 6.

Table 6 – Structure of «Order» Object Extension for Tobacco Industry

Field	Description	Type	Mandatory
factoryId	Factory identifier (Global location number)	String	Yes
factoryName	Factory name	String	No
factoryAddress	Factory address	String	No
factoryCountry	Factory country	String	Yes
productionLineId	Production line identifier	String	Yes
productCode	Goods code (SKU)	String	Yes
productDescription	Product description	String	Yes
poNumber	Production order number	String	No
expectedStartDate	Start date of production under this order	String (yyyy-mm-dd)	No

Example of REST request (for tobacco industry) can be found in Figure 7.

```
POST /api/v2/tobacco/orders?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Content-Length: 718
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Host: localhost:8080

{
  "products": [ {
    "gtin": "01334567894339",
    "quantity" : 20,
    "serialNumberType" : "OPERATOR",
    "templateId" : 3}
  ],
  "factoryId": "Identifier",
  "factoryName": "Tobacco Fac",
  "factoryAddress": "Address",
  "factoryCountry": "Country",
  "productionLineId": "1",
  "productCode": "6789",
  "productDescription": "Simple ",
  "poNumber": "12345",
  "expectedStartDate": "2019-03-01"
}
```

Example of REST request (for tobacco industry)

Figure 7

4.4.1.1.2 Extensions for <Footwear> goods group

Description of <Order> object extension for the <Footwear> goods group can be found in Table 7.

Table 7 – Description of <Order> object extension for <Footwear> goods group

Field	Description	Type	Mandatory
contactPerson	Contact person	String	Yes
releaseMethodType	Method of goods introduction into circulation	String (See section 5.3.1.1)	Yes
createMethodType	IM creation method	String (See section 5.3.1.3)	Yes
productionOrderId	Production order identifier	String	No
country	Country of production Catalog value coded in line with international classification of the world countries (ISO 3166-1)	String (See section 5.3.1.14)	No

Example of REST request for the <Footwear> goods group is shown in Figure 8.

```
POST /api/v2/shoes/orders?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Content-Length: 783
Host: localhost:8080

{
  "products" : [ {
    "gtin" : "01334567894339",
    "quantity" : 20,
    "serialNumberType": "OPERATOR",
    "templateId": 1,
  }],
  "contactPerson": "Ivanov P.A.",
  "releaseMethodType": "PRODUCTION",
  "createMethodType": "SELF_MADE",
  "productionOrderId": "08528091-808a-41ba-a55d-d6230c64b332",
  "country": "KZ"
}
```

Example of REST Request for light industry - <Footwear> Goods Group Category

Figure 8

4.4.1.1.3 Extensions for «Alcohol» goods group

Description of «Order» object extension for the «Alcohol» goods group can be found in Table 8.

Table 8 – Description of «Order» object extension for «Alcohol» goods group category

Field	Description	Type	Mandatory
contactPerson	Contact person	String	Yes
releaseMethodType	Method of goods introduction into circulation	String (See section 5.3.1.1)	Yes
createMethodType	IM creation method	String (See section 5.3.1.3)	Yes
productionOrderId	Production order identifier	String	No
country	Country of production Catalog value coded in line with international classification of the world countries (ISO 3166-1)	String (See section 5.3.1.14)	No

Description of «OrderProduct» object extension for «Alcohol» goods group can be found in Table 9.

Table 9 – Description of «OrderProduct» object extension for «Alcohol» goods group category

Field	Description	Type	Mandatory
cisType	Marking code type. Valid values: — UNIT – Goods unit; — GROUP – Group consumer packing	String (see section 5.3.1.12)	Yes

Example of REST request for the «Alcohol» goods group is shown in Figure 9.

```
POST /api/v2/alcohol/orders?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Content-Length: 783
Host: localhost:8080

{
  "products" : [ {
    "gtin" : "01334567894339",
    "quantity" : 20,
    "serialNumberType": "OPERATOR",
    "templateId": 13,
    "cisType": "UNIT"
  }],
  "contactPerson": "Ivanov P.A.",
  "releaseMethodType": "PRODUCTION",
  "createMethodType": "SELF_MADE",
  "productionOrderId": "08528091-808a-41ba-a55d-d6230c64b332",
  "country": "KZ"
}
```

Example of REST request for the «Alcohol» goods group

Figure 9

4.4.1.1.4 Extensions for pharmaceutical industry

Description of ‹Order› object extension for pharmaceutical industry can be found in Table 10.

Table 10 – Structure of ‹Order› object extension for pharmacological industry

Field	Description	Type	Mandatory
factoryId	Factory identifier (Global location number)	String	Yes
factoryName	Factory name	String	No
factoryAddress	Manufacturer address	String	No
factoryCountry	Factory country	String	Yes
productionLineId	Production line identifier	String	Yes
productCode	Goods code (SKU)	String	Yes
productDescription	Product description	String	Yes
poNumber	Production order number	String	No
expectedStartDate	Start date of production under this order	String (yyyy-mm-dd)	No
releaseMethodType	Method of goods introduction into circulation	String (See section 5.3.1.1)	Yes
country	Country of production Catalog value coded in line with international classification of the world countries (ISO 3166-1)	String (See section 5.3.1.14)	No

Example of REST request (for pharmaceutical industry) is shown in Figure 10.

```
POST /api/v2/pharma/orders?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Content-Length: 718
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Host: localhost:8080

{
  "products": [ {
    "gtin": "01334567894339",
    "quantity": 20,
    "serialNumberType": "OPERATOR",
    "templateId": 5
  ],
  "factoryId": "Identifier",
  "factoryName": "Pharma Factory",
  "factoryAddress": "Address",
  "factoryCountry": "Country",
  "productionLineId": "1",
  "productCode": "6789",
  "productDescription": "Simple ",
  "poNumber": "12345",
  "expectedStartDate": "2019-03-01",
  "releaseMethodType": "PRODUCTION",
  "country": "KZ"
}
```

Example of REST request (for pharmaceutical industry)

Figure 10

4.4.1.1.5 Extensions for dairy industry

Description of «Order» object extension for dairy industry can be found in Table 11.

Table 11 – Structure of «Order» Object Extension for Dairy Industry

Field	Description	Type	Mandatory
releaseMethodType	Method of goods introduction into circulation	String (See section 5.3.1.1)	Yes
createMethodType	IM creation method	String (See section 5.3.1.3)	Yes
productionOrderId	Production order identifier	String	No
country	Country of production catalog value coded in line with international classification of the world countries (ISO 3166-1)	String (See section 5.3.1.14)	No

Description of «OrderProduct» object extension for the «Dairy products» goods group can be found in Table 12.

Table 12 – Description of «OrderProduct» Object Extension for «Dairy products» Goods Group Category

Field	Description	Type	Mandatory
cisType	Marking code type. Available values: — UNIT – Goods unit; — GROUP – Group packing.	String (See section 5.3.1.12)	Yes

Example of REST request (for dairy industry) is shown in Figure 11.

```
POST /api/v2/milk/orders?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Content-Length: 718
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Host: localhost:8080

{
  "products": [
    {
      "gtin": "01334567894339",
      "quantity": 20,
      "serialNumberType": "OPERATOR",
      "cisType": "UNIT",
      "templateId": 20
    }
  ],
  "releaseMethodType": "PRODUCTION",
  "createMethodType": "SELF_MADE",
  "productionOrderId": "08528091-808a-41ba-a55d-d6230c64b332",
  "country": "KZ"
}
```

Example of REST request (for dairy industry)

Figure 11

4.4.1.1.6 Extensions for «Items of Clothing, Bed, Table, Bath and Kitchen Linen» Goods Group

Description of «Order» object extension for «Items of Clothing, Bed, Table, Bath and Kitchen Linen» goods group can be found in Table 11.

Table 13 – Structure of «Order» Object Extension for “Items of Clothing, Bed, Table, Bath and Kitchen Linen” Goods Group

Field	Description	Type	Mandatory
releaseMethodType	Method of goods introduction into circulation	String (See section 5.3.1.1)	Yes
createMethodType	IM creation method	String (See section 5.3.1.3)	Yes
productionOrderId	Production order identifier	String	No
country	Country of production Catalog value coded in line with international classification of the world countries (ISO 3166-1)	String (See section 5.3.1.14)	No

Description of «OrderProduct» object extension for «Items of clothing, bed, table, bath and kitchen linen» goods group can be found in Table 14.

Table 14 – Description of «OrderProduct» Object Extension for «Items of Clothing, Bed, Table, Bath and Kitchen Linen» Goods Group

Field	Description	Type	Mandatory
cisType	Marking code type. Available values: — UNIT – Goods unit;	String (See section 5.3.1.12)	Yes

Example of REST request (for <Items of Clothing, Bed, Table, Bath and Kitchen Linen> goods group) can be found in Figure 12.

```
POST /api/v2/lp/orders?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Content-Length: 718
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Host: localhost:8080

{
  "products": [ {
    "gtin": "01334567894339",
    "quantity": 20,
    "serialNumberType": "OPERATOR",
    "cisType": "UNIT",
    "templateId": 10
  }],
  "releaseMethodType": "PRODUCTION",
  "createMethodType": "SELF_MADE",
  "productionOrderId": "08528091-808a-41ba-a55d-d6230c64b332",
  "country": "KZ"
}
```

Example of REST request (for <Items of Clothing, Bed, Table, Bath and Kitchen Linen> goods group)

Figure 12

4.4.1.1.7 Extensions for «Packaged water and sugar-containing beverages» goods group

Description of «Order» object extension for the «Packaged water and sugar-containing beverages» goods group can be found in Table 15.

Table 15 – Structure of «Order» Object Extension for «Packaged water and sugar-containing beverages» Goods Group

Field	Description	Type	Mandatory
releaseMethodType	Method of goods introduction into circulation	String (See section 5.3.1.1)	Yes
createMethodType	IM creation method	String (See section 5.3.1.3)	Yes
productionOrderId	Production order identifier	String	No
country	Country of production Catalog value coded in line with international classification of the world countries (ISO 3166-1)	String (See section 5.3.1.14)	No

Description of «OrderProduct» object extension for the «Packaged water and sugar-containing beverages» goods group can be found in Table 16.

Table 16 – Description of «OrderProduct» Object Extension for «Packaged water and sugar-containing beverages» Goods Group

Field	Description	Type	Mandatory
cisType	Marking code type. Available values: — UNIT – Goods unit; — GROUP – Group packing.	String (See section 5.3.1.12)	Yes

Example of REST request (for «Packaged water and sugar-containing beverages» goods group) can be found in Figure 13.

```
POST /api/v2/water/orders?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Content-Length: 718
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Host: localhost:8080

{
  "products": [ {
    "gtin": "01334567894339",
    "quantity": 20,
    "serialNumberType": "OPERATOR",
    "cisType": "UNIT",
    "templateId": 16
  }],
  "releaseMethodType": "PRODUCTION",
  "createMethodType": "SELF_MADE",
  "productionOrderId": "08528091-808a-41ba-a55d-d6230c64b332",
  "country": "KZ"
}
```

Example of REST request (for «Packaged water and sugar-containing beverages» goods group)

Figure 13

4.4.1.2. Request Response

The method returns a unique identifier of the order and planned order completion time in milliseconds (divide the result time by 1,000 for seconds, and by 60 for minutes). “orderId” value is used to obtain MC from the order upon its completion (See section 4.4.6). For error codes, see subsection 6.2.

Table 17 – Format of Response to Request

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
orderId	Unique ID of the MC emission order	String (UUID)	Yes
expectedCompleteTimestamp	Order expected completion time, in milliseconds	Integer (\$int64)	Yes

Example of JSON response is shown in Figure 14.

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Content-Length: 111
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId" : "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "orderId" : "b024ae09-ef7c-449e-b461-05d8eb116c79",
  "expectedCompleteTimestamp" : 5100
}
```

Example of JSON Response

Figure 14

4.4.2. Method <Send a report on MC dropout/rejection>

This method is used to send a report on MC dropout/rejection to the OMS. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP-header with the name <clientToken>. If the security marker (token) has been obtained by calling <Get security marker by username and password> method (See section 4.4.13), user name must be passed in HTTP header with "userName" name.

Note: At the moment, this method is only available for <Tobacco products> and <Medicines for human use>, <Dairy products> goods groups.

4.4.2.1. Request

Structure of JSON request to send a report on MC dropout/rejection to the OMS.

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/dropout?omsId={omsId}`
Method: POST
Content-type: application/json
clientToken: {clientToken}
userName: {userName}

Request string parameters are given in Table 18.

Table 18 – Request String Parameters

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes

Description of <DropoutReport> object structure for sending of the report on MC dropout/rejection to the OMS can be found in Table 19.

Table 19 – <DropoutReport> Object Structure

Field	Description	Type	Mandatory
dropoutReason	Withdrawal reason	String (See section 5.3.1.11)	Yes
sntins	Information on dropped out MCs (full marking code)	JSON Array of String	Yes

Note: Quantity of the MCs in the disposal report shall not exceed 30,000 codes.

4.4.2.1.1 Extensions for tobacco industry

Description of <DropoutReport> object extension for tobacco industry can be found in Table 20.

Table 20 – Description of <DropoutReport> Object Extension for Tobacco Industry

Field	Description	Type	Mandatory
sourceDocDate	Document date	String (yyyy-mm-dd)	No
sourceDocNum	Identifier of the document on which the writing-off is based	String	No
address	Address of writing-off	String	Yes
withChild	Indicator for writing-off of all nested elements	Boolean (Default value=false)	Yes
participantId	Tax Identification Number	String	Yes
productionOrderId	Production order identifier	String	No
productionLineId	Production line identifier	String	No

Notes:

- If sourceDocDate and sourceDocNum fields are not filled in, the OMS will automatically fill them in with the following values:
 - sourceDocDate is the current date in unixTime UTC:0 in milliseconds;
 - sourceDocNum is the current date in unixTime UTC:0 in milliseconds.

Example of REST request (for tobacco industry) can be found in Figure 15.

```
POST /api/v2/tobacco/dropout?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 168

{
  "dropoutReason": "DEFECT",
  "sntins": [ "SNTIN1", "SNTIN2" ],
  "sourceDocNum": "12345",
  "sourceDocDate": "2018-05-01",
  "address": "г. Алматы, ул. Назарбаева, 71",
  "withChild": false,
  "participantId": "3543033591",
  "productionOrderId": "123",
  "productionLineId": "7098"
}
```

Example of REST request (for tobacco industry)

Figure 15

4.4.2.1.2 Extensions for pharmaceutical industry

Description of «DropoutReport» object extension for pharmaceutical industry can be found in Table 21.

Table 21 – Description of «DropoutReport» Object Extension for Pharmaceutical Industry

Field	Description	Type	Mandatory
sourceDocDate	Document date	String (yyyy-mm-dd)	No
sourceDocNum	Identifier of the document on which the writing-off is based	String	No
address	Address of writing-off	String	Yes
withChild	Indicator for writing-off of all nested elements	Boolean (Default value=false)	Yes
participantId	Tax Identification Number	String	Yes
productionOrderId	Production order identifier	String	No
productionLineId	Production line identifier	String	No

Notes:

- If sourceDocDate and sourceDocNum fields are not filled in, the OMS will automatically fill them in with the following values:
 - sourceDocDate is the current date in unixTime UTC:0 in milliseconds;
 - sourceDocNum is the current date in unixTime UTC:0 in milliseconds.

Example of REST request (for pharmaceutical industry) is shown in Figure 16.

```
POST /api/v2/pharma/dropout?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 168

{
  "dropoutReason": "DEFECT",
  "sntins": [ "SNTIN1", "SNTIN2" ],
  "sourceDocNum": "12345",
  "sourceDocDate": "2018-05-01",
  "address": "Г. Алматы, ул. Назарбаева, 71",
  "withChild": false,
  "participantId": "3543033591",
  "productionOrderId": "123",
  "productionLineId": "7098"
}
```

Example of REST request (for pharmaceutical industry)

Figure 16

4.4.2.1.3 Extensions for dairy industry

Description of «DropoutReport» object extension for dairy industry can be found in Table 22.

Table 22 – Description of «DropoutReport» Object Extension for Dairy Industry

Field	Description	Type	Mandatory
sourceDocDate	Document date	String (yyyy-mm-dd)	No
sourceDocNum	Identifier of the document on which the writing-off is based	String	No
withChild	Indicator for writing-off of all nested elements	Boolean (Default value=false)	Yes
participantId	Tax Identification Number	String	Yes

Notes:

1. If sourceDocDate and sourceDocNum fields are not filled in, the OMS will automatically fill them in with the following values:

- sourceDocDate is the current date in unixTime UTC:0 in milliseconds;
- sourceDocNum is the current date in unixTime UTC:0 in milliseconds.

Example of REST request (for dairy industry) is shown in Figure 17.

```
POST /api/v2/milk/dropout?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
Content-Length: 168

{
  "dropoutReason": "DEFECT",
  "sntins": ["SNTIN1", "SNTIN2"],
  "sourceDocNum": "12345",
  "sourceDocDate": "2018-05-01",
  "withChild": false,
  "participantId": "3543033591"
}
```

Example of REST request (for dairy industry)

Figure 17

4.4.2.2. Request Response

Upon successful completion of request, the server returns HTTP code 200 and unique identifier of the report on MC dropout/rejection assigned by OMS. The received identifier of the report on MC dropout/rejection is used to obtain the report processing status (See section 4.4.10). Structure of the response to the request to send the aggregation information can be found in Table 23. For error codes, see subsection 6.2.

Table 23 – Format of Response to Request to Send Report on MC Dropout / Rejection

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
reportId	Unique identifier of the report on MC dropout / rejection (OMS)	String (UUID)	Yes

Example of JSON response is shown in Figure 18.

```

HTTP/1.1 200 OK
Content-Length: 74
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId" : "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "reportId" : "46795d19-5024-404e-9275-959ac89ccb57"
}

```

Example of JSON Response

Figure 18

4.4.3. Method <Send Report on MC Aggregation>

This method is used to send the report on MC aggregation to the OMS. In the report, MCs are sent without verification code. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP-header with the name “clientToken”. If the security marker (token) has been obtained by calling “Get security marker by username and password” method (See section 4.4.13), user name must be passed in HTTP header with “userName” name.

Note: At the moment this method is available for all goods groups, except for <Packaged water and sugar-containing beverages> and <Items of Clothing, Bed, Table, Bath and Kitchen Linen>.

4.4.3.1. Request

The structure of JSON request to send report on MC aggregation to the OMS.

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/aggregation?omsId={omsId}`
Method: POST
Content-type: application/json
clientToken: {clientToken}
userName: {userName}

Request string parameters are given in Table 24.

Table 24 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes

Description of <AggregationReport> object structure of the request to send information on aggregation is given in Table 26.

Table 25 – Structure of <AggregationReport> Object

Field	Description	Type	Mandatory
aggregationUnits	Array of aggregation units	Array of AggregationUnit (Table 26)	Yes
participantId	Tax Identification Number	String	Yes

Description of <AggregationUnit> object structure is given in Table 26.

Table 26 – Structure of <AggregationUnit> Object

Field	Description	Type	Mandatory
aggregatedItemCount	Actual quantity (in pcs) per aggregation unit	Integer (\$int32)	Yes
aggregationType	Aggregation type	String (See section 5.3.1.6)	Yes
aggregationUnitCapacity	Package unit capacity	Integer (\$int32)	Yes
sntins	Array of aggregated MCs Format in cases when in the MC — GS1 standard is used: — 01+goods nomenclature (GTIN)+21+serial number — GS1 standard is not used: — goods nomenclature (GTIN)+serial number	JSON Array of String	Yes
unitSerialNumber*	Identification code of aggregation unit	String	Yes

Notes:

1. For “Dairy Products” goods group:
 - “unitSerialNumber” (aggregate code) field may contain:
 - shipping package code (SSCC - length of 20 numeric symbols with GS1 AI 00 application identifier);
 - shipping package code of the format determined by the circulation participant (from 20 to 74 characters long containing digits and letters of the Latin alphabet, as well as special characters A-Z a-z 0-9 % & ' « () * + , - _ . / : ; < = > ? !);
 - group package identification code.
 - “sntins” (goods code) field may contain:
 - consumer package identification code;
 - group package identification code;
 - shipping package code (SSCC - length of 20 numeric symbols with GS1 AI 00 application identifier);
 - shipping package code of the format determined by the circulation participant (from 20 to 74 characters long containing digits and letters of the Latin alphabet, as well as special characters A-Z a-z 0-9 % & ' « () * + , - _ . / : ; < = > ? !).
 - SSCC shipping packages may be aggregated in the SSCC shipping package.
 - The shipping packages codes of the format determined by the circulation participant (from 20 to 74 characters long) may be aggregated in the SSCC shipping package.
 - Group packages may be aggregated in the SSCC shipping package.

- Consumer packages may be aggregated in the SSCC shipping package.
- SSCC shipping packages may be aggregated into the packages of the format determined by the circulation participant (from 20 to 74 characters long).
- The shipping packages codes of the format determined by the circulation participant (from 20 to 74 characters long) may be aggregated in the packages of the format determined by the circulation participant (from 20 to 74 characters long).
- Group packages may be aggregated into the packages of the format determined by the goods circulation participant (from 20 to 74 characters long).
- Consumer packages may be aggregated into the packages of the format determined by the circulation participant (from 20 to 74 characters long).
- Consumer packages may be aggregated in the group packages.

2. For “Alcohol” goods group:

- “unitSerialNumber” (aggregate code) field may contain:
 - shipping package code (SSCC - length of 18 numeric symbols);
 - group package identification code.
- “sntins” (goods code) field may contain:
 - consumer package identification code;
 - group package identification code;
 - shipping package code (SSCC - length of 18 numeric symbols).
- SSCC shipping packages may be aggregated in the SSCC shipping package.
- Group packages may be aggregated in the SSCC shipping package.
- Consumer packages may be aggregated in the SSCC shipping package.
- Consumer packages may be aggregated in the group packages.

3. For “Footwear” goods group:

- “unitSerialNumber” (aggregate code) field may contain:
 - shipping package code (SSCC - length of 18 numeric symbols).
- “sntins” (goods code) field may contain:
 - consumer package identification code;
 - shipping package code (SSCC - length of 18 numeric symbols).
- SSCC shipping packages may be aggregated in the SSCC shipping package.
- Consumer packages may be aggregated in the SSCC shipping package.

4. For “Medicines for human use” goods group:

- “unitSerialNumber” (aggregate code) field may contain:
 - shipping package code (SSCC - length of 20 numeric symbols with GS1 AI 00 application identifier).
- “sntins” (goods code) field may contain:
 - consumer package identification code;
 - shipping package code (SSCC - length of 20 numeric symbols with GS1 AI 00 application identifier).
- SSCC shipping packages may be aggregated in the SSCC shipping package/

— Consumer packages may be aggregated in the SSCC shipping package.

5. For “Tobacco products” goods group:

— “unitSerialNumber” (aggregate code) field may contain:

- shipping package code (SSCC - length of 20 numeric symbols with GS1 AI 00 application identifier);
- shipping package code of the format determined by the circulation participant (more than 20 characters long containing digits and letters of the Latin alphabet, as well as special characters A-Z a-z 0-9 % & ' « () * + , - _ . / : ; < = > ? !).
- group package code.

— “sntins” (goods code) field may contain:

- consumer package identification code;
- group package identification code;
- shipping package code (SSCC - length of 20 numeric symbols with GS1 AI 00 application identifier);
- shipping package code of the format determined by the circulation participant (more than 20 characters long containing digits and letters of the Latin alphabet, as well as special characters A-Z a-z 0-9 % & ' « () * + , - _ . / : ; < = > ? !).

— SSCC shipping packages may be aggregated in the SSCC shipping package.

— The shipping packages codes of the format determined by the circulation participant (more than 20 characters long) may be aggregated in the SSCC shipping package.

— Group packages may be aggregated in the SSCC shipping package.

— Consumer packages may be aggregated in the SSCC shipping package.

— SSCC shipping packages may be aggregated into the packages of the format determined by the circulation participant (more than 20 characters long).

— The shipping packages codes of the format determined by the circulation participant (more than 20 characters long) may be aggregated in the packages of the format determined by the circulation participant (more than 20 characters long).

— Group packages may be aggregated into the packages of the format determined by the circulation participant (more than 20 characters long).

— Consumer packages may be aggregated into the packages of the format determined by the circulation participant (more than 20 characters long).

— Consumer packages may be aggregated in the group packages.

4.4.3.1.1 Extensions for tobacco industry

Description of <AggregationReport> object extension for tobacco industry can be found in Table 27.

Table 27 – Description of “AggregationReport” Object Extension for Tobacco Industry

Field	Description	Type	Mandatory
productionLineId	Production line identifier	String	Yes
productionOrderId	Production order identifier	String	No

Example of REST request for tobacco industry is shown in Figure 19.

```
POST /api/v2/tobacco/aggregation?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Content-Length: 785
Host: localhost:8080

{
  "participantId": "3543033591",
  "productionLineId": 235431,
  "productionOrderId": 123,
  "aggregationUnits": [
    {
      "aggregatedItemCount" : 2,
      "aggregationType" : "AGGREGATION",
      "aggregationUnitCapacity" : 10,
      "sntins" : [
        "00000000666999QbUMR5M",
        "00000000666999r2Aw4Ge"],
      "unitSerialNumber" : "0004601234560000010"
    }
  ]
}
```

Example of REST request for tobacco Industry

Figure 19

4.4.3.1.2 Extensions for pharmaceutical industry

Description of <AggregationReport> object extension for pharmaceutical industry can be found in Table 28.

Table 28 – Description of “AggregationReport” Object Extension for Pharmaceutical Industry

Field	Description	Type	Mandatory
productionLineId	Production line identifier	String	Yes
productionOrderId	Production order identifier	String	No

Example of REST request for pharmaceutical industry is shown in Figure 20.

```
POST /api/v2/pharma/aggregation?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Content-Length: 785
Host: localhost:8080

{
  "participantId": "3543033591",
  "productionLineId": 235431,
  "productionOrderId": 123,
  "aggregationUnits": [
    {
      "aggregatedItemCount" : 2,
      "aggregationType" : "AGGREGATION",
      "aggregationUnitCapacity" : 10,
      "sntins" : [
        "00000000666999QbUMR5M",
        "00000000666999r2Aw4Ge"
      ],
      "unitSerialNumber" : "00046012345600000010"
    }
  ]
}
```

Example of REST request for pharmaceutical industry

Figure 20

4.4.3.2. Request Response

Upon successful completion of request, the server returns HTTP code 200 and the unique identifier of the report assigned by OMS. The received identifier of the report on MC aggregation is used to get the reports processing status (See section 4.4.10). Structure of the response to the request to send the aggregation information can be found in Table 29. For error codes, see subsection 6.2.

Table 29 – Format of Response to Request to Send Information on Aggregation

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
reportId	Unique identifier of the OMS report	String (UUID)	Yes

Example of JSON response is shown in Figure 21.

```

HTTP/1.1 200 OK
Content-Length: 74
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId" : "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "reportId" : "fab1c0e4-9590-4ed7-8d58-18862d6a9aab"
}

```

Example of JSON Response

Figure 21

4.4.4. Method <Send Report on MC Utilization (Application)>

This method is used to send a report on MC use to the OMS. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP header with the name <clientToken>. If the security marker (token) has been obtained by calling <Get security marker by username and password> method (See section 4.4.13), user name must be passed in HTTP header with "userName" name.

Note: At the moment, this method is available for <Tobacco products>, <Alcohol> and <Medicines for human use>, <Dairy products>, <Packaged water and sugar-containing beverages> goods groups.

4.4.4.1. Request

The structure of JSON request to send report on MC usage to the OMS.

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/utilisation?omsId={omsId}`
Method: POST
Content-type: application/json
clientToken: {clientToken}
userName: {userName}

Request string parameters are given in Table 30.

Table 30 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes

Description of <UtilisationReport> object structure for sending reports on MC utilization to the OMS is given in Table 31.

Note: Transmitted marking codes used as <sntins> parameters shall include a complete marking code. The quantity of MC in the utilization report shall not exceed 30,000 codes.

Table 31 – Structure of “UtilisationReport” Object

Field	Description	Type	Mandatory
sntins	String array (full marking code)	JSON Array of String	Yes
usageType	Type of use	String (See section 5.3.1.9)	Yes

Note: Only VERIFIED value is allowable in “Type of Utilization” parameter for <Dairy products> and <Packaged water and sugar-containing beverages> goods groups.

4.4.4.1.1 Extensions for tobacco industry

Description of <UtilisationReport> object extension for tobacco industry can be found in Table 32.

Table 32 – Description of <UtilisationReport> Object Extension for Tobacco Industry

Field	Description	Type	Mandatory
productionLineId	Production line identifier	String	Yes
productionOrderId	Production order identifier	String	No
brandcode	Product brandname	String (256)	No
sourceReportId	Identifier of a report on APCS application	String (36)	No

Example of REST request for tobacco industry is shown in Figure 22.

```
POST /api/v2/tobacco/utilization?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Content-Length: 145
Host: localhost:8080

{
  "sntins" : [ "SNTIN1", "SNTIN2" ],
  "usageType" : "PRINTED",
  "productionLineId" : "1",
  "productionOrderId": "123",
  "brandcode" : "2212Brandcode",
  "sourceReportId": "8ed74f90-0119-48f2-b289-379707934e2f"
}
```

Example of REST request for tobacco Industry

Figure 22

4.4.4.1.2 Extensions for pharmaceutical industry

Description of «UtilisationReport» object extension for pharmaceutical industry can be found in Table 33.

Table 33 – Description of «UtilisationReport» Object Extension for Pharmaceutical Industry

Field	Description	Type	Mandatory
seriesNumber	Production series number	String (1-256)	Yes
expirationDate	Expiration date. Date indicated in accordance with GOST ISO 8601–2001. Date format: YYYY-MM-DD	String (yyyy-mm-dd)	Yes
productionOrderId	Production order identifier	String	No
brandcode	Product brandname	String (256)	No
sourceReportId	Identifier of a report on APCS application	String (36)	No

Example of REST request for pharmaceutical industry is shown in Figure 23.

```
POST /api/v2/pharma/tilization?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Content-Length: 145
Host: localhost:8080

{
  "sntins" : [ "SNTIN1", "SNTIN2" ],
  "usageType" : "PRINTED",
  "expirationDate": "2019-03-01",
  "seriesNumber": "123",
  "productionOrderId": "123",
  "brandcode" : "2212Brandcode",
  "sourceReportId": "8ed74f90-0119-48f2-b289-379707934e2f"
}
```

Example of REST request for pharmaceutical industry

Figure 23

4.4.4.1.3 Extensions for dairy industry

Description of «UtilisationReport» object extension for dairy industry can be found in Table 34.

Table 34 – Description of «UtilisationReport» Object Extension for Dairy Industry

Field	Description	Type	Mandatory
seriesNumber	Production series number	String (1-256)	Yes
expirationDate	Expiration date. Date indicated in accordance with GOST ISO 8601–2001. Date format: YYYY-MM-DD	String (yyyy-mm-dd)	Yes

Example of REST request for dairy industry is shown in Figure 24.

```
POST /api/v2/milk/tilization?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Content-Length: 145
Host: localhost:8080

{
  "sntins" : [ "SNTIN1", "SNTIN2" ],
  "usageType" : "VERIFIED",
  "expirationDate": "2019-03-01",
  "seriesNumber": "123"
}
```

Example of REST request for dairy industry

Figure 24

4.4.4.1.4 Extensions for «Packaged water and sugar-containing beverages» goods group

Description of «UtilisationReport» object extension for the «Packaged water and sugar-containing beverages» goods group can be found in Table 35.

Table 35 – Description of «UtilisationReport» Object Extension for «Packaged water and sugar-containing beverages» goods group

Field	Description	Type	Mandatory
sourceReportId	Identifier of a report on APCS application	String (36)	No

Example of REST request for «Packaged water and sugar-containing beverages» goods group can be found in Figure 25.

```
POST /api/v2/water/utilization?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Content-Length: 145
Host: localhost:8080

{
  "sntins" : [ "SNTIN1", "SNTIN2" ],
  "usageType" : "VERIFIED",
  "sourceReportId": "8ed74f90-0119-48f2-b289-379707934e2f"
}
```

Example of REST request for «Packaged water and sugar-containing beverages» goods group

Figure 25

4.4.4.2. Request Response

Upon successful completion of request, the server returns HTTP code 200 and unique identifier of the report on MC utilization assigned by the OMS. The received identifier of the report on MC utilization is used to obtain the report processing status (See section 4.4.10). Structure of the response to the request for sending of the report on utilization is given in Table 36. For error codes, see subsection 6.2.

Table 36 – Format of Response to Request to Send Report on MC Application

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
reportId	Unique identifier of the MC application report (the OMS)	String (UUID)	Yes

Example of JSON response is shown in Figure 26.

```
HTTP/1.1 200 OK
Content-Length: 74
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId" : "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "reportId" : "3179f5d2-2bf5-47d1-8df0-9452b257d851"
}
```

Example of JSON Response

Figure 26

4.4.5. Method <Close suborder/order>

This method is used to close MC array (suborder) by using the following parameters: security marker (token), OMS identifier, order identifier and GTIN. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP-header with the name “clientToken”.

Note: Suborder - MC array in one GTIN in the order, after closing the last suborder, the order will be closed automatically.

4.4.5.1. Request

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/buffer/close?orderId={orderId}>in={gtin}&omsId={omsId}&lastBlockId={lastBlockId}`

Method: POST

clientToken: {clientToken}

Request string parameters are given in Table 37.

Table 37 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
orderId	Identifier of the order for MC emission in the OMS	String (UUID)	Yes
gtin	GTIN of the goods by which the MC issue must be stopped	String (14) [0-9]{14}	No
lastBlockId	Identifier of last code block received (default value: 0)	String	No

Notes:

1. In the request, in order to confirm the closing of suborder, it is necessary to send <lastBlockId> parameter, which shall include the value of the latest code block identifier received in the response message when calling the method <Obtain MC from the order> (See section 4.4.6). If the marking codes have not been requested by the circulation participant, it is not necessary to fill in <lastBlockId> field (“0” will be set by default).
2. Goods code (GTIN) is required only if the suborder is closed. If it is missing in the request parameters, all suborders of the current order will be closed.

Example of REST request is shown in Figure 27.

```
POST /api/v2/tobacco/buffer/close? orderId=b024ae09-ef7c-449e-b461-  
05d8eb116c79&gtin=01334567894339&lastBlockId=0&omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1  
Accept: application/json  
Content-Type: application/x-www-form-urlencoded; charset=UTF-8  
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f  
Host: localhost:8080
```

Example of REST Request

Figure 27

4.4.5.2. Request Response

Upon successful request completion, the server returns HTTP code 200 and unique identifier to the OMS. For the structure of response to request to close the suborder by GTIN specified, see Table 38. For error codes, see subsection 6.2.

Table 38 – Format of Response to Request to Close Suborder by GTIN Specified

Field	Description	Type
omsId	Unique OMS identifier. (UUID)	String

Example of JSON response is shown in Figure 28.

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Content-Length: 19
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId" : "CDF12109-10D3-11E6-8B6F-0050569977A1"
}
```

Example of JSON Response

Figure 28

4.4.6. Method «Obtain MC from the order»

This method is used to obtain the MC array of the order by using the following parameters: security marker (token), OMS identifier, order identifier, GTIN, quantity of the requested codes. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP header with the name «clientToken».

4.4.6.1. Request

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/codes?omsId={omsId}&orderId={orderId}>in={gtin}&quantity={quantity}&lastBlockId={lastBlockId}`

Method: GET

clientToken: {clientToken}

Request string parameters are given in Table 39.

Table 39 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
orderId	Identifier of MC emission order	String (UUID)	Yes
gtin	GTIN of goods used for requesting the codes	String (14) [0-9]{14}	Yes
quantity	Quantity of requested codes	Integer (\$int32)	Yes
lastBlockId	Code block identifier issued in the previous request. May be equal to 0 in the first MC request from pool. Further, previous packet identifier shall be sent. Default value: 0	String	No

Notes:

1. A circulation participant receives the emitted marking codes (guaranteed obtainment of the emitted marking codes) and sends the confirmation of marking code receipt in the requests and when closing the order. In this case
 - in the first request for marking codes, the value of «lastBlockId» attribute shall be «0» (zero), the response message will contain a code block identifier («blockId» attribute value) which must be specified in the next request for marking codes, and each subsequent request shall contain «lastBlockId» attribute value equal to the code block identifier received in the previous request (transmission of code block identifier confirms that the emitted marking codes were received);

— closing the order is the final step, which is executed automatically when printing the last MCs. Manual closing of the order is performed optionally by the circulation participant, when all MCs have not been printed (See section 4.4.5). In the request for closing (`<lastBlockId>` attribute), the last received code block identifier must be transmitted if part of the MC has already been printed. If MC has not been printed in the order, `<lastBlockId>` attribute does not need to be filled in (“0” value will be set).

Example of REST request is shown in Figure 29.

```
GET /api/v2/tobacco/codes?orderId=b024ae09-ef7c-449e-b461-
05d8eb116c79&gtin=01334567894339&quantity=15&lastBlockId=0&omsId=CDF12109-10D3-11E6-8B6F-
0050569977A1 HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Host: localhost:8080
```

Example of REST Request

Figure 29

4.4.6.2. Request Response

Upon successful request completion, the server returns HTTP code 200 and MC array. For the format of response to request for MC obtainment for the goods specified, see Table 40. For error codes, see subsection 6.2.

Table 40 – Format of Response to Request for MC for the Goods Specified

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
codes	MC array	JSON Array of Strings	Yes
blockId	MC package identifier	String (UUID)	Yes

Example of JSON response is shown in Figure 30.

```

HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Length: 820
Content-Type: application/json; charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId" : "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "codes" : [ "010460165303004621\u003drxDV3M\u001d93VXQI", ... ],
  "blockId" : "012cc7b0-c9e4-4511-8058-2de1f97a87b0"
}

```

Example of JSON Response

Figure 30

4.4.7. Method <Get a status of MC array from the order>

This method is used the following parameters (to get the current status of the MC array from the order): security marker (token), OMS identifier, <orderId> order identifier and GTIN. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP header with the name <clientToken>.

4.4.7.1. Request

Structure of the request for obtaining a status of MC array from the order.

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/buffer/status?omsId={omsId}&orderId={orderId}>in={gtin}`

Method: GET

clientToken: {clientToken}

Request string parameters are given in Table 41.

Table 41 – Request String Parameters

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
orderId	Identifier of MC emission order	String (UUID)	Yes
gtin	GTIN of the goods, the order status of which is required	String (14) [0-9]{14}	Yes

Example of REST request is shown in Figure 31.

```
GET /api/v2/tobacco/buffer/status?orderId=b024ae09-ef7c-449e-b461-05d8eb116c79&gtin=01334567894339&omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Host: localhost:8080
```

Example of REST Request

Figure 31

4.4.7.2. Request Response

Format of JSON response to request to obtain the MC array status can be found in Table 42. For error codes, see subsection 6.2.

Table 42 – Format of Response to Request, object “BufferInfo”

Field	Description	Type	Mandatory
availableCodes	The total quantity of available MC for goods in buffer and pools	Integer (\$int32)	Yes
bufferStatus	Buffer status	String (See section 5.3.1.7)	Yes
gtin	GTIN on which the request was made	String (14) [0-9]{14}	Yes
leftInBuffer	Number of unused MCs. (local buffer)	Integer (\$int32)	Yes
omsId	Unique OMS identifier.	String	Yes
orderId	Unique identifier of the MC emission order. Order on which the request was made	String (UUID)	Yes
poolInfos	Array of pools created for the buffer	JSON Array of PoolInfo Object (Note: If the order is rejected, this field will contain the value “Order declined: “ with the reason for order decline Table 43)	No
poolsExhausted	MC pools exhausted	Boolean	Yes
rejectionReason	Reason for rejection of the buffer by the OMS	String	No
totalCodes	Ordered quantity of MCs in the order	Integer (\$int32)	Yes
totalPassed	Total quantity of the MCs received from the buffer	Integer (\$int32)	Yes
unavailableCodes	Quantity of unavailable codes	Integer (\$int32)	Yes
expiredDate	MC expiration date Format: UnixTime	String (dd.mm.yyyy)	No

Note: If the order is rejected, this field will contain the value “Order declined: “ with the reason for order decline

Table 43 – <PoolInfo> Object Format

Field	Description	Type	Mandatory
isRegistrarReady	Pool readiness	Boolean	Yes
lastRegistrarErrorTimestamp	Time stamp of the last observed pool error	Long (\$int64)	Yes
leftInRegistrar	Quantity of MC left in pool	Integer (\$int32)	Yes
quantity	Ordered quantity of MC in pool	Integer (\$int32)	Yes
registrarErrorCount	Number of pool errors	Integer (\$int32)	Yes
registrarId	Pool identifier (number)	String	Yes
rejectionReason	Rejection reason	String	No
status	MC pool status.	String (see section 5.3.1.5)	Yes

Example of JSON response is shown in Figure 32.

```

HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Length: 659
Content-Type: application/json; charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "poolInfos" : [ {
    "status" : "READY",
    "quantity" : 9,
    "leftInRegistrar" : 0,
    "registrarId" : "Virtual Registrar",
    "isRegistrarReady" : true,
    "registrarErrorCount" : 0,
    "lastRegistrarErrorTimestamp" : 0
  }, {
    "status" : "READY",
    "quantity" : 11,
    "leftInRegistrar" : 0,
    "registrarId" : "Virtual Registrar",
    "isRegistrarReady" : true,
    "registrarErrorCount" : 0,
    "lastRegistrarErrorTimestamp" : 0
  } ],
  "leftInBuffer" : 0,
  "totalCodes" : 20,
  "poolsExhausted": false,
  "unavailableCodes" : 0,
  "availableCodes" : 20,
  "orderId" : "b024ae09-ef7c-449e-b461-05d8eb116c79",
  "gtin" : "01334567894339",
  "bufferStatus" : "ACTIVE",
  "totalPassed": 0,
  "omsId" : "CDF12109-10D3-11E6-8B6F-0050569977A1"
}

```

Example of JSON Response

Figure 32

4.4.8. Method <Get Orders Status>

This method is used to get the orders status by using the following parameters: security marker (token), OMS identifier. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP header with the name <clientToken>.

Notes:

1. The method is intended for APCS recovery after a total data loss; use of its functional capabilities during normal operating process with the OMS is prohibited.
2. This method (as well as the method of creating orders) can be addressed from one source not more than 100 times per second.

4.4.8.1. Request

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/orders?omsId={omsId}`
Method: GET
clientToken: {clientToken}

Request string parameter is given in Table 44.

Table 44 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes

Example of REST request is shown in Figure 33.

```
GET /api/v2/tobacco/orders?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Host: localhost:8080
```

Example of REST Request

Figure 33

4.4.8.2. Request Response

Upon successful request completion, the server returns HTTP code 200, data on orders status and unique identifier of the OMS. For the format of response to request to get the aggregation content, see Table 45. For error codes, see subsection 6.2.

Table 45 – Format of Response to Request for Order Status

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
orderInfos	Array of orders with their statuses	JSON Array of OrderSummaryInfo (Table 46)	Yes

Table 46 – <OrderSummaryInfo> Object Format

Field	Description	Type	Mandatory
orderId	Identifier of MC emission order	String (UUID)	Yes
orderStatus	Order status	String (See section 5.3.1.10)	Yes
buffers	Array of information on buffer status	JSON Array of BufferInfo (Table 42)	Yes
createdTimestamp	Order creation time	Integer (\$int64)	Yes
declineReason	Reason for order decline	String	No

Example of JSON response is shown in Figure 34.

```

HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Length: 953
Content-Type: application/json; charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
{
  "omsId" : "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "orderInfos" : [ {
    "orderId" : "b024ae09-ef7c-449e-b461-05d8eb116c79",
    "orderStatus" : "READY",
    "createdTimestamp" : 1550650989568,
    "buffers" : [ {
      "poolInfos" : [ {
        "status" : "READY",
        "quantity" : 9,
        "leftInRegistrar" : 0,
        "registrarId" : "Virtual Registrar",
        "isRegistrarReady" : true,
        "registrarErrorCount" : 0,
        "lastRegistrarErrorTimestamp" : 0
      }, {
        "status" : "READY",
        "quantity" : 11,
        "leftInRegistrar" : 0,
        "registrarId" : "Virtual Registrar",
        "isRegistrarReady" : true,
        "registrarErrorCount" : 0,
        "lastRegistrarErrorTimestamp" : 0
      } ],
      "leftInBuffer" : 20,
      "totalCodes" : 20,
      "unavailableCodes" : 0,
      "orderId" : "b024ae09-ef7c-449e-b461-05d8eb116c79",
      "gtin" : "01334567894339",
      "bufferStatus" : "ACTIVE",
      "omsId" : "CDF12109-10D3-11E6-8B6F-0050569977A1"
    } ]
  } ]
}

```

Example of JSON Response

Figure 34

4.4.9. Method <Receive aggregation information>

This method is used to get the aggregation content by using the following parameters: token, OMS identifier, aggregate identifier. The token is generated by the OMS during the OMS client registration. The token is sent to the server within HTTP header with the name <clientToken>.

4.4.9.1. Request

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/aggregation/info?omsId={omsId} & unitSerialNumber={unitSerialNumber}`

Method: GET

Accept: application/json

clientToken: {clientToken}

Request string parameter is given in Table 47.

Table 47 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
unitSerialNumber	Aggregate identifier. Since the code can contain the special characters, the value must be converted into the actual format of ASCII (URL Encoding)	String	Yes

Example of REST request is shown in Figure 35.

```
GET /api/v2/tobacco/aggregation/info?unitSerialNumber=01000000077799921311SMYX800510000&omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
```

Example of REST Request

Figure 35

4.4.9.2. Request Response

Upon successful request completion, the server returns HTTP code 200 and data of the goods nomenclature catalog. For the format of response to request to get the aggregation content, see Table 48. For error codes, see subsection 6.2.

Table 48 – Format of Response to Request for Aggregation Information, object <AggregationInfo>

Field	Description	Type	Mandatory
aggregationUnits	Array of aggregation units	Array of AggregationUnit (Table 26)	Yes
omsId	Unique OMS identifier.	String (UUID)	Yes
participantId	Tax Identification Number	String	Yes
productsInfo	Product information	Array of ProductInfo (Table 49)	No

Description of <ProductInfo> object structure is given in Table 49.

Table 49 – <ProductInfo> Object Format

Field	Description	Type	Mandatory
gtin	GTIN of the product	String (14) [0-9]{14}	Yes
name	Goods name	String	Yes

Example of JSON response is shown in Figure 36.

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
Content-Length: 119
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8

{
  "aggregationUnit": {
    "aggregatedItemsCount": 48,
    "aggregationType": "AGGREGATION",
    "aggregationUnitCapacity": 50,
    "sntins": [
      "0100000848839984215LJ",
      "0100000848839984215Py"
    ],
    "unitSerialNumber": "01000000077799921311SMYX8005100000"
  },
  "omsId": "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "participantId": "string",
  "productsInfo": [
    {
      "gtin": "string",
      "name": "string"
    }
  ]
}
```

Example of JSON Response

Figure 36

4.4.9.2.1 Extension for tobacco industry

Description of `<AggregationInfo>` object extension for manufacturers of tobacco industry can be found in Table 50.

Table 50 – Description of `<AggregationInfo>` Object Extension for Manufacturers of Tobacco Industry

Field	Description	Type	Mandatory
productionLineId	Production line identifier	String	Yes
productionOrderId	Production order identifier	String	No

Example of JSON response is shown in Figure 37.

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
Content-Length: 119
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8

{
  "aggregationUnit": {
    "aggregatedItemCount": 48,
    "aggregationType": "AGGREGATION",
    "aggregationUnitCapacity": 50,
    "sntins": [
      "0100000848839984215LJ",
      "0100000848839984215Py"
    ],
    "unitSerialNumber": "010000000777999213l1SMYX8005100000"
  },
  "omsId": "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "participantId": "string",
  "productionLineId": 235431,
  "productionOrderId": 123,
  "productsInfo": [
    {
      "gtin": "string",
      "name": "string"
    }
  ]
}
```

Example of JSON Response

Figure 37

4.4.9.2.2 Extension for pharmaceutical industry

Description of «AggregationInfo» object extension for manufacturers of pharmaceutical industry can be found in Table 51.

Table 51 – Description of «AggregationInfo» Object Extension for Manufacturers of Pharmaceutical Industry

Field	Description	Type	Mandatory
productionLineId	Production line identifier	String	Yes
productionOrderId	Production order identifier	String	No

Example of JSON response is shown in Figure 38.

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
Content-Length: 119
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8

{
  "aggregationUnit": {
    "aggregatedItemCount": 48,
    "aggregationType": "AGGREGATION",
    "aggregationUnitCapacity": 50,
    "sntins": [
      "0100000848839984215LJ",
      "0100000848839984215Py"
    ],
    "unitSerialNumber": "01000000077799921311SMYX8005100000"
  },
  "omsId": "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "participantId": "string",
  "productionLineId": 235431,
  "productionOrderId": 123,
  "productsInfo": [
    {
      "gtin": "string",
      "name": "string"
    }
  ]
}
```

Example of JSON Response

Figure 38

4.4.10. Method <Obtain the Report Processing Status>

This method is used to get the report processing status by using the following parameters: security marker (token), OMS identifier, report identifier. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP header with the name <clientToken>.

4.4.10.1. Request

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/report/info?omsId={omsId} & reportId={reportId}`

Method: GET

clientToken: {clientToken}

Request string parameter is given in Table 52.

Table 52 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
reportId	Unique identifier of the OMS report	String (UUID)	Yes

Example of REST request is shown in Figure 39.

```
GET /api/v2/tobacco/report/info?reportId=fab1c0e4-9590-4ed7-8d58-18862d6a9aab&omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json; charset=UTF-8
Host: localhost:8080
```

Example of REST Request

Figure 39

4.4.10.2. Request Response

Upon successful request completion, the server returns HTTP code 200, OMS unique identifier and the report processing status. For the format of response to request to get the report processing status, see Table 53. For error codes, see subsection 6.2.

Table 53 – Format of Response to Request for Report Processing Status

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
reportId	Unique identifier of the OMS report	String (UUID)	Yes
reportStatus	Report processing status	String (See section 5.3.1.8)	Yes
errorReason	Reason for report rejection (detected error)	String	No (it fills in only if reportStatus is “REJECTED”)

Example of JSON response is shown in Figure 40.

```

HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
Content-Length: 108
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId": "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "reportId": "fab1c0e4-9590-4ed7-8d58-18862d6a9aab",
  "reportStatus": "SENT"
}

```

Example of JSON Response

Figure 40

4.4.11. Method <Send APCS log files>

This method is used to send APCS log files in zip format and uses the following parameters: security marker (token), OMS identifier and zip file. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP header with the name <clientToken>.

4.4.11.1. Request

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/logs?omsId={omsId}`

Method: POST

Content-: multipart/form-data

clientToken: {clientToken}

Request string parameter is given in Table 54.

Table 54 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes

Request parameter can be found in Table 55.

Table 55 – Request Body Parameters

Parameter	Description	Type	Mandatory
log	zip file	-	Yes

Example of request is shown in Figure 41.

```
POST /api/v2/tobacco/logs/upload?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-type: multipart/form-data; charset=UTF-8; boundary=6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Host: localhost:8080
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=omsId
123456
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm
Content-Disposition: form-data; name=log; filename=logs.zip
Content-Type: text/plain
Test data
--6o2knFse3p53ty9dmcQvWAIx1zInP11uCfbm-
```

Request Example

Figure 41

4.4.11.2. Request Response

Upon successful request completion, the server returns HTTP code 200 and unique identifier to the OMS. The format of response to request for OMS availability can be found in Table 56. For error codes, see subsection 6.2.

Table 56 – Format of Response to Request for OMS Availability

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes

Example of JSON response is shown in Figure 42.

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Content-Length: 19
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
{
  "omsId" : "CDF12109-10D3-11E6-8B6F-0050569977A1"
}
```

Example of JSON Response

Figure 42

4.4.12. Method <Check OMS availability>

This method is used to check the availability of the OMS and uses the following parameters: security marker (token) and OMS identifier. The security marker (token) is generated by OMS during the OMS client registration. Security marker (token) is sent to the server within HTTP header with the name <clientToken>.

4.4.12.1. Request

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/ping?omsId={omsId}`
Method: GET
clientToken: {clientToken}

Request string parameter is given in Table 57.

Table 57 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String(UUID)	Yes

Example of REST request is shown in Figure 43.

```
GET /api/v2/tobacco/ping?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1 HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Host: localhost:8080
```

Example of REST Request

Figure 43

4.4.12.2. Request Response

Upon successful request completion, the server returns HTTP code 200 and unique identifier to the OMS. The format of response to request for OMS availability can be found in Table 58. For error codes, see subsection 6.2.

Table 58 – Format of Response to Request for OMS Availability

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes

Example of JSON response is shown in Figure 44.

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Content-Length: 19
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId": "CDF12109-10D3-11E6-8B6F-0050569977A1"
}
```

Example of JSON Response

Figure 44

4.4.13. Method <Get security marker by username and password>

This method allows to get the security marker (token) by username and password. The method uses the following parameters: OMS identifier, username and password. Received security marker (token) is used in HTTP header with <clientToken> name when calling other methods (address to other resources) of the OMS API.

Note: This method is supported only for the OMS installed in the infrastructure of the goods circulation participant.

4.4.13.1. Request

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/token?omsId={omsId} & username={username} & password={password}`

Method: GET

tokenName: tokenName

Accept: application/json

HTTP Header Parameters can be found in Table 59.

Table 59 – HTTP Header Parameters

Parameter	Description	Type	Mandatory
tokenName	Username (name of the client device)	String	Yes

Request string parameters are given in Table 60.

Table 60 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
username	User's first name	String	Yes
password	User password	String	Yes

Example of REST request is shown in Figure 45.

```
GET /api/v2/tobacco/token?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1
&username=admin&password=password HTTP/1.1
Accept: application/json
Host: localhost:8080
```

Example of REST Request

Figure 45

4.4.13.2. Request Response

Upon successful request completion, the server returns HTTP code 200, unique identifier of the OMS and security marker (token). The format of response to request for OMS availability can be found in Table 61. For error codes, see subsection 6.2.

Table 61 – Format of Response to Request for Security Marker

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
token	Security marker	String	Yes

Example of JSON response is shown in Figure 46.

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Content-Length: 19
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId": "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "token": "1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f",
}
```

Example of JSON Response

Figure 46

4.4.14. Method “Obtain OMS and API version”

This method helps obtaining the versions of OMS and API. The method does not require parameters.

4.4.14.1. Request

REST request parameters

URL: `http://<server-name>[:server-port]/api/v2/{extension}/version`

Method: GET

Accept: application/json

4.4.14.2. Request Response

Upon successful completion of request, the server returns HTTP code 200, OMS version number and OMS API number. The format of response to request for OMS availability can be found in Table 62. For error codes, see subsection 6.2.

Table 62 – Format of Response to Request for OMS and API Version

Field	Description	Type	Mandatory
apiVersion	OMS API version	String	Yes
omsVersion	OMS version	String	Yes

Example of JSON response is shown in Figure 47.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 19

{
  "apiVersion": "2.0.0.54",
  "omsVersion": "3.1.8.0"
}
```

Example of JSON Response

Figure 47

4.4.15. Method “Obtain the list of marking code package identifiers”

This method is used to obtain the list of marking code package identifiers issued previously from the marking code order that are necessary for repeated request for marking codes by calling method <Re-obtain marking codes from marking code order> (See section 4.4.16).

The method uses the following parameters: OMS identifier, order identifier, GTIN. Security marker (token), which is generated by the OMS when registering the OMS client, is sent to the server in HTTP header with name <clientToken>.

4.4.15.1. Restrictions

The list of previously issued marking code package identifiers may be obtained only if the marking code suborder has not been closed and the first print request has been processed via API.

4.4.15.2. Request

Parameters of REST request:

URL: `http://<server-name>[:server-port]/api/v2/{extension}/codes/blocks?omsId={omsId} & orderId={orderId}>in={gtin}`

Method: `GET`

clientToken: `{clientToken}`

Accept: `application/json`

Table 63 – Request String Parameters

Parameter	Description	Type	Mandatory
omsId	Unique OMS identifier.	String(UUID)	Yes
orderId	Marking code order identifier. String value. Identifier value in accordance with ISO/IEC 9834-8. Template: [0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}	String (36) (UUID)	Yes
gtin	Goods code (GTIN) for which marking code package identifiers are requested. Template: [0-9]{14}	String (14) [0-9]{14}	Yes

Example of request is shown in Figure 48.

```
GET /api/v2/{extension}/codes/blocks?omsId=CDF12109-10D3-11E6-8B6F-0050569977A1&orderId=b024ae09-e
ef7c-449e-b461-05d8eb116c79&gtin=01334567894339 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Host: localhost:8080
```

Request Example

Figure 48

4.4.15.3. Request Response

Upon successful request completion, the server returns HTTP code 200, a list of marking code package identifiers. For the format of response to request for the list of marking code package identifiers for the specified order of marking and goods codes, see Table 64. For error codes, see subsection 6.2.

Table 64 – Format of Response to Request for the List of MC Packet Identifiers for the Marking and Goods Code Order Specified

Field	Description	Type	Mandatory
orderId	Marking code order identifier. String value. Identifier value in accordance with ISO/IEC 9834-8. Template: [0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}	String (36) (UUID)	Yes
gtin	Goods code (GTIN) for which marking code package identifiers are requested. Template: [0-9]{14}	String (14) [0-9]{14}	Yes
omsId	Unique OMS identifier.	String (UUID)	Yes
blocks	List of marking code packages	Array of Block objects (Table 65)	Yes

Table 65 – Format of MC Packet List, Object <Block>

Field	Description	Type	Mandatory
blockId	Marking code package identifier sent within the request. String value. Identifier value in accordance with ISO/IEC 9834-8. Template: [0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}	String (36)(UUID)	Yes
blockDateTime	Date, time of marking code package creation. Format: UnixTime	Integer (\$int64)	Yes
quantity	Quantity of marking codes in marking code package	Integer (\$int32)	Yes

Example of response is shown in Figure 49.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
X-RequestId: 1a59cc06-4371-4981-9e9e-019b435bfa72
Content-Length: 310

{
  "orderId": "b024ae09-ef7c-449e-b461-05d8eb116c79",
  "omsId": "CDF12109-10D3-11E6-8B6F-0050569977A1",
  "gtin": "01334567894339",
  "blocks": [
    {
      "blockId": "a024ae09-ef7c-449e-b461-05d8eb116c90",
      "blockDateTime": 1573986891,
      "quantity": 100
    },
    {
      "blockId": "a024ae09-ef7c-449e-b461-05d8eb116c93",
      "blockDateTime": 1573986910,
      "quantity": 100
    }
  ]
}
```

Response Example

Figure 49

4.4.16. Method <Re-obtain marking codes from marking code order>

This method is used to re-obtain array of the emitted MCs from the marking code suborder in case when marking codes were not obtained due to communication errors or errors on the side of the System interacting with the OMS.

The method uses the following parameters: order identifier, GTIN, marking code package identifier. Security marker (token), which is generated by the OMS when registering the OMS client, is sent to the server in HTTP header with name <clientToken>.

The list of previously issued marking code package identifiers is obtained by calling method <Obtain the list of marking code package identifiers> (See section 4.4.15).

4.4.16.1. Restrictions

Marking codes may be requested again only if:

- 1) they have been requested previously via API;
- 2) marking code suborder has not been closed.

4.4.16.2. Request

See request parameters below:

URL: `http://<server>[:port]/api/v2/{extension}/codes/retry?orderId={orderId}>in={gtin}&blockId={blockId}`

Method: GET

clientToken: {clientToken}

Accept: application/json

Table 66 – Request String Parameters

Parameter	Description	Type	Mandatory
orderId	Marking code order identifier. String value. Identifier value in accordance with ISO/IEC 9834-8. Template: [0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}	String (36) (UUID)	Yes
gtin	Goods code (GTIN) for which marking codes are requested again. Template: [0-9]{14}	String (14) [0-9]{14}	Yes
blockId	Code block identifier. String value. Identifier value in accordance with ISO/IEC 9834-8. Template: [0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}	String (36) (UUID)	Yes

Example of request is shown in Figure 50.

```
GET /api/v2/{extension}/codes/retry?orderId=b024ae09-ef7c-449e-b461-
05d8eb116c79&gtin=01334567894339&blockId=a024ae09-ef7c-449e-b461-05d8eb116c90 HTTP/1.1
Accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Host: localhost:8080
```

Request Example

Figure 50

4.4.16.3. Request Response

Upon successful request completion, the server returns HTTP code 200 and marking code package. Format of response to request for re-obtainment of MC for the goods specified can be found in Table 67. For error codes, see subsection 6.2.

Table 67 – Format of Response to Request for Re-obtainment of MC for the Goods Specified

Field	Description	Type	Mandatory
omsId	Unique OMS identifier.	String (UUID)	Yes
codes	Marking code package	Strings array (JSON Array of Strings)	Yes
blockId	Marking code package identifier sent within the request. String value. Identifier value in accordance with ISO/IEC 9834-8. Template: [0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}	String (36) (UUID)	Yes

Example of response is shown in Figure 51.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 310

{
  "omsId": "bb179f1f-d6d5-4e09-8012-2a28648474e9",
  "codes": ["010460165303004621\u003drxDV3M\u001d93VXQI", "..."],
  "blockId": "a024ae09-ef7c-449e-b461-05d8eb116c90"
}
```

Response Example

Figure 51

4.4.17. Method “Receive receipt by unique document identifier”

This method is required to obtain receipts by unique document identifier (order, report) and uses the following parameters: security marker (token), OMS identifier, document identifier (order or report).

4.4.17.1. Request

See REST request parameters below:

URL: `http://<server-name>[:server-port]/api/v2/{extension}/receipts?docId={docId}&omsId={omsId}`

Method: GET

clientToken: {clientToken}

Table 68 – Request String Parameters

Parameter	Description	Type	Mandatory
docId	Unique identifier of the document (order or report)	String (UUID)	Yes
omsId	Unique OMS identifier.	String (UUID)	Yes

Example of request is shown in Figure 52.

```
GET /api/v2/tobacco/receipts/omsId=CDF12109-10D3-11E6-8B6F-0050569977A1&docId=05f52b01-ba4b-4dc7-a94a-7846db44ac63 HTTP/1.1
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Accept: application/json;
```

Request Example

Figure 52

4.4.17.2. Response

Upon successful request completion, the server returns HTTP code 200 and the array of the receipts corresponding to the document specified in the parameters. The structure of response to request is given in the table below (Table 69). For error codes, see subsection 6.2.

Table 69 – Format of Response to Request

Field	Description	Type	Mandatory
receipts	Array of desired receipts (Table 70)	Array of Receipt JSON objects	Yes

Table 70 – Format of Receipt Object

Field	Description	Type	Mandatory
content	Signed receipt content	String (string)	Yes
signature	Operator's EQES	String (string)	Yes

Example of response is shown in Figure 53.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
X-RequestId: 9a59aa06-4371-4980-9e9e-019b435bab89
{
  "receipts": [
    {
      "content": "{\"id\": \"89e00089-658d-4f55-ac4c-ebe27e7b39bf\", \"sourceDocId\": \"4bb1dd92-d544-4d78-b005-c2e7c08492b9\", \"sourceDocHash\": \"71127ced\", \"destinationId\": \"89e00089-658d-4f55-ac4c-ebe27e7b39bf\", \"destination\": \"OMS\", \"docType\": \"REPORT_UTILIZE_RESERV\", \"result\": \"ACCEPTED\", \"participantId\": \"600000078\", \"transactionId\": \"b6a3dbbd-3ea2-4cef-a54c-ff467a3726b5\", \"createdTimestamp\": 1596290531159, \"metadata\": {\"@class\": \"com.equiron.sitemanager.ap.i.gisdb.model.metadata.ReportMetadata\", \"reportId\": \"b6a3dbbd-3ea2-4cef-a54c-ff467a3726b5\", \"productGroup\": \"tobacco\", \"amount\": 4.00, \"quantity\": 4}}",
      "signature": "MIIL8gYJKoZIhvcNAQcCoIIL4zCCC98CAQExDjAMBggqhQMHAQECAgUAMAsGCSqGSIB3DQEHAaCCCU8wgg1LMIII+KADAgECAhEB05
      KoAK6q0KFHdgDG3Xi/8DAKBggqhQMHAQEDAjCCAegxGzAZBqkqhkiG9w0BCQEWGDNhQHN1cnR1bS5ydTEYMBYGBSqFA2QBEg0xMTE2N
      jczMDA4NTM5MRowGAYIKoUDA4EDAQESDDAwNjY3MzI0MDMyODELMAkGA1UEBhMCU1UxMzAxBgNVBAgMKjY2INCh0LLQtGA0LTQu9C+
      0LLRgdC60LDRjyDQvtCx0LvQsNGB0YLrjDEhMB8GA1UEBwwY0JXQutCw0YLQtdGA0LjQvdCx0YPRgNCzMViwUAYDVQQJDEnRg9C70Lj
      RhtCwINCj0LvRjNGP0L3QvtCy0YHQutCw0Y8sINC0LiAxMywg0LvQuNGC0LXRgCDQkCwg0L7RhNC40YegMjA5INCRMlwagYDVQQKDGP
      PQtCx0YnQtdGB0YLQstC+INGBINC+0LPRgNCw0L3QuNGH0LXQvdC90L7QuSDQvtGC0LLQtGA0YHRgtCy0LXQvdC90L7RgdGC0YzRj
      iAi0KHQtGA0YLrg9C8LdCf0YDQviIxDbqBgnVBAMMY9Ce0LHRidC10YHRgtCy0L4g0Yeg0L7Qs9GA0LDQvdC40YfQtdC90L3QvtC5
      INC+0YLQstC10YLrgdGC0LLQtGA0L3QvtGB0YLrjNGOICLQodC10YDRgtGD0Lwt0J/RgNC+IjAeFw0xOTA4MTkxMDA4NDZaFw0yMDA
      5MTAxMjAwMDFaMIIBAjEaMBgGCCqFAwOBAwEBEgwMDc3MzEzNzY4MTIxGDAWBgUqhQNkARINMTE3Nzc0NjcyNTkyNTERMCKGA1UECg
      wi0J7QntCeICLQntCf0JXQoNCQ0KLQntCgLdCm0KDQn9CiIjEuMCwGA1UECQw10J/QoC3QmtCiINCc0JjQoNCQLCDQ1NCe0JwgNiwg0
      J7QpCAyNTEVMBMGA1UEBwwM0JzQvtGB0LrQstCwMRwwGgYDVQQIDBM3NyDQsy4g0JzQvtGB0LrQstCwMQswCQYDVQQGEwJSVTeRMCkG
```

```

A1UEAwwi0J7QntCeICLQntCf0JXQoNCQ0KLQntCgLdCm0KDQn9CiIjBmMB8GCCqFAwcBAQEGBMGByqFAwICJAAGCCqFAwcBAQICA0M
ABEAuXci13efe1HKUWVJ0bBIhbUc8Jc11WXoA57QknSglBU2Sd/ExdmxPwskHARRwvsd6myZAIwZ1IxwhSf/joKo4IFVjCCBVIwDg
YDVR0PAQH/BAQDAgTwMBMGA1UdIAQMAowCAYGKoUDZHEBMDYGA1UdJQvMC0GCCsGAQUFBwMCBgchQMCaiIGBgcqhQMDgTkBBgcqh
QMDBwgBBgYqhQNkAgIwgUGCCsGAQUFBwEBBIHIMHFMDcGCCsGAQUFBzABhitodHRwOi8vcGtpLnNlcnR1bS1wcm8ucnUvb2NzcHeY
MDEyL29jc3Auc3JmMEYGCCsGAQUFBzACHjpodHRwOi8vY2Euc2VydHvtLXByby5yds9jZXJ0aWZpY2F0ZXMc2VydHvtLXByby1xLTi
wMTkuY3J0MEIGCCsGAQUFBzACHjZodHRwOi8vY2Euc2VydHvtLnJ1L2N1cnRpZm1jYXR1cy9ZXJ0dW0tchJvLXEtMjAxOS5jcnQwKw
YDVR0QBCQwIoAPMjAxOTA4MTkxMDA4NDVagQ8yMDIwMDkxMDEyMDAwMVowggEzBgUqhQNkASCASgwggEkDCsi0JrRgNC40L/RgtC+0
J/RgNC+IENTUCIgKNCy0LXRgNGB0LjRjyA0LjApDFMi0KPQtNC+0YHRgtC+0LLQtdGA0YRgtC+0LLQtdGA0YHQuNC4IDIuMAXP0KHQt
dGA0YLQuNGE0LjQutCw0Yig0YHQvtC+0YLQstC10YLrgdGC0LLQuNGPIOKE1iDQodCkLzEyNC0zMzgwINC+0YIgMTEuMDUuMjAxOA
xP0KHQt
dGA0YLQuNGE0LjQutCw0Yig0YHQvtC+0YLQstC10YLrgdGC0LLQuNGPIOKE1iDQodCkLzEyNC0zNTkyINC+0YIgMTCuMTAuMjAx
0DA2BgUqhQNkbwQtDCsi0JrRgNC40L/RgtC+0J/RgNC+I
ENTUCIgKNCy0LXRgNGB0LjRjyA0LjApMHcGA1UdHwRwMG4wN6A1oDOGMWh0dHA6Ly9jYS5zZXJ0dW0tchJvLnJ1L2Nkc9zZXJ0dW0t
chJvLXEtMjAxOS5jcmwwM6AxoC+GLWh0dHA6Ly9jYS5zZXJ0dW0ucnUvY2RwL3N1cnR1bS1wcm8tcS0yMDE5LmNybDCBggYHKoUDAgI
xAgR3MHUwZRZAaHR0cHM6Ly9jYS5rb250dXIucnUvYWJvdXQvZG9jdw1bnRzL2NyeXB0b3Byby1saWN1bnN1LXF1YwxpZm11Zawd0K
HQmtCRINCa0L7QvdGC0YPRgCDQuCDQ1NCX0J4DagXgBAzVK/pkyf0Q1jv0qLswggFgBgvNHSMEEgFXMIIIBU4AuxNzWhk4mQZ0wTg+1L
1MRuoIWf40hggEspIIBKDCCASQxHjAcBqkqhkig9w0BCQEWD2RpdEBtaW5zdnlhei5ydTELMAkGA1UEBhMCU1UxGDAwBgvNVBAgMDzc3
INCc0L7RgdC60LLQsDEZMBCGA1UEBwwQ0LMuINCc0L7RgdC60LLQsDEuMCwGA1UECQwI0YPQu9C40YbQsCDQotCy0LXRgNGB0LrQsNG
PLCDQtNC+0LwgNzEsMCoGA1UECgj0JzQuNC90LrQvtC80YHQstGP0LfrjCDQoNC+0YHRgdC40LgxGDAwBgvQhQNkARINMTA0Nzcmj
AyNjcwMTEaMBgGCCqFAwOBAwEBEgwwMDc3MTA0NzQzNzUxDaqBgvNBAMMI9Cc0LjQvdC60L7QvNGB0LLRj9C30Ywg0KDQvtGB0YHQu
NC4ggsAjnaQdQAAAACVDAdBgNVHQ4Ef gQUtBvCnh3xN92kRCoxJxy7+vJRh0wCgYIKoUDBwEBAwIDQQAZXEvYKoU0+jpyKE2jmy6o
wMsFP20a4Dqm0jSIgi0onZ0Zvn9YbfN/9Qm5ZkAjqSS5IKFbs95H1hIIQRI18mc4MYICaDCCAmQCAQEWggH/MIIB6DEbMBkGCSqGSIB
3DQEJARYMY2Fac2VydHvtLnJ1MRgwFgYFKoUDZAESDExMTY2NzMwMDg1MzKxGjAYBggqhvQMDgQMBARIMMDA2NjczMjQwMzI4MQswCQ
YDVQQGEwJSVTEzMDEGA1UECAwqNjYg0KHQstC10YDQtNC70L7QstGB0LrQsNGPINC+0LHQu9Cw0YHRgtGMMSEwHwYDVQQHDBjQ1dC60
LDRgtC10YDQuNC90LHRg9GA0LMxUjBQBgNVBAkMSdGD0LrQuNGG0Lag0KPQu9GM0Y/QvdC+0LLRgdC60LDRjyw0LQuIDEzLCDQu9C4
0YLQtdGAINCQLCDQvtGE0LjRgSAyMDkg0JExbDBqBgvNBVAoMY9Ce0LHRidC10YHRgtCyoL4g0Yeg0L7Qs9GA0LDQvdC40YfQtdC90L3
QvtC5INC+0YLQstC10YLrgdGC0LLQtdC90L3QvtGB0YLrjNGOICLQodC10YDRgtGD0Lwt0J/RgNC+IjFsmGoGA1UEAwxj0J7QsdGj0L
XRgdGC0LLQviDRgSDQvtCz0YDQsNC90LjRh9C10L3QvdC+0Lkg0L7RgtCyoLXRgtGB0YLQstC10L3QvdC+0YHRgtGM0Y4gItCh0LXRg
NGC0YPQvC3Qn9GA0L4iAhEB05KoAK6q0KFHdgDG3Xi/8DAMBggqhvQMAQECAGuAMAwGCCqFAwcBAQMcbQAEQMA4advvMR2TtIMUB3ck
z+xvgx5EkCnBna5ws15noZzROE4Fs1dIbeMI2UYQnfYIfgj6WaPtX1j7P8FpzlyE/ms=”

}
]
}

```

Response Example

Figure 53

5. INPUT AND OUTPUT DATA

5.1. Character, organization and preliminary preparation of input and output data

5.1.1. Data sources

Main data sources for the system are:

- 1) Allied information systems used in information and communication, as well as functional exchange.
- 2) Data entered by system users.

Operations for:

- 1) Analog-to-digital and digital-to-analog conversion of signals;
- 2) Optical symbol recognition;
- 3) Other operations to make the information acceptable for processing in the computer are not provided as functions of OMS-Cloud 3.1 AS.

5.1.2. Methods for arranging collection, transmission, monitoring and correction of the information

Information arrays are collected during the system operation as follows:

- 1) Receiving (by means of interaction services) the structured XML documents formed on the basis of specified XSD schemes.
- 2) Entering the information in the screen forms by users followed by its saving in the database.

Data integrity monitoring is ensured by application software and utilities (restrictions, indexes, primary and secondary keys) built in the used DBMS. Data shall be entered and corrected only through program components of the system. Direct access to the database is not provided for the users.

Main requirements for the processes of information collection, sending, monitoring and modification are ensuring reliability, verifiability, confidentiality, availability, up-to-date status of collected and sent data.

Reliability requirement means arrangement of the information collection and sending processes in such a way that the sent and collected information is not misrepresented.

Verifiability requirement means arrangement of the information collection and sending processes in such a way as to enable monitoring of the sent information reliability.

Confidentiality requirement means granting access to the information in strict compliance with the established priorities and access control rules.

Availability requirement means the possibility to obtain the collected information and to send it in principle.

Up-to-date status requirement means arrangement of the information collection and sending processes in such a way as to ensure the available information sending within a reasonable period for its analysis.

5.2. Format, Description and Method of Encoding Input and Output Data When Using API

Format, description and method of encoding input and output data when using API are given in the description of relevant methods.

5.3. Catalogs Available Through API

5.3.1. Catalogs to operate with the marking codes

5.3.1.1. Catalog <Method of Goods Release into Circulation>

For the list of possible values of Catalog <Method of Goods Release into Circulation>, see Table 71.

Table 71 – Possible Values of Catalog <Method of Goods Release into Circulation>

Constant	Value	Type
"PRODUCTION",	Manufacture in Kazakhstan	String
IMPORT	Imported into Kazakhstan (Import)	String
REMAINS	Remains marking (available only for "Footwear", "Clothing Items, Bed, Table, Bath and Kitchen Linen" goods groups)	String
REMARK	Remarking (available only for the "Footwear" goods group)	String
COMMISSION	Accepted for commission from an individual (available only for "Footwear", "Clothing Items, Bed, Table, Bath and Kitchen Linen" goods groups)	String

5.3.1.2. Catalog <Method of Individual Serial Number Generation>

For the list of possible values of Catalog <Method of Individual Serial Number Generation>, see Table 72.

Table 72 – Possible Values of Catalog <Method of Individual Serial Number Generation>

Constant	Value	Type
SELF_MADE	No assistance	String
OPERATOR	By Operator	String

5.3.1.3. Catalog <Manufacturing method>

For the list of possible values of Catalog "Manufacturing method", see Table 73.

Table 73 – Possible Values of Catalog <Manufacturing method>

Constant	Value	Type
SELF_MADE	No assistance	String
CEM	LMC	String
CM	Contract manufacturing	String
CL	Logistic warehouse	String

5.3.1.4. Catalog <MC Templates>

For the list of possible values of Catalog “MC Templates”, see Table 74.

Table 74 – Possible Values of Catalog <MC Template>

Constant	Value	Type
1	01 + gtin + 21 + serial (13 chars)	String
3	01 + gtin + 21 + serial (7 chars)	String
4	gtin + serial (7 chars)	String
5	01 + gtin + 21 + serial (13 chars)	String
10	01 + gtin + 21 + serial (13 chars)	String
13	01 + gtin + 21 + serial (7 chars)	String
16	01 + gtin + 21 + serial (13 chars)	String
17	01 + gtin + 21 + serial (13 chars)	String
20	01 + gtin + 21 + serial (6 chars)	String

Note:

- 1) When generating serial numbers independently, their length for the template of dairy products - templateld=20 shall consist of 5 characters. When marking codes are emitted, the serial number will consist of 6 characters including the country code. The country code is indicated by the Emission Server and shall be entered before the received serial number.
- 2) templateld=2 is excluded for “Medicines for human use” goods group, this template is no longer applicable for emitting the marking codes, as well as the MCs emitted by templateld=2 cannot be specified in the reports on utilization (application). The template with code “5” shall be used instead.
- 3) For the templates of "Footwear" (templateld=1), "Clothing Items, Bed, Table, Bath and Kitchen Linen" (templateld=10), "Packaged water and sugar-containing beverages" (templateld=16) goods groups, in case of independent generation method, the serial number length shall consist of 12 characters. When marking codes are emitted, the serial number will consist of 13 characters including the country code. The country code is indicated by the Emission Server and shall be entered before the received serial number.

Table 75 – Description of MC Templates

Name	Description
Template 1	Footwear
Template 3	Cigarettes, cartons
Template 4	Cigarettes, packs
Template 5	Medicines

Name	Description
Template 10	Clothing items, bed, table, bath and kitchen linens
Template 13	Alcohol. Consumer package
Template 16	Packaged water and sugar-containing beverages
Template 17	Alcohol. Group consumer packing
Template 20	Dairy products. Consumer and Group consumer packing

Note: Template of the cigarette pack is characterized by the lack of AI in the template and in the MC.

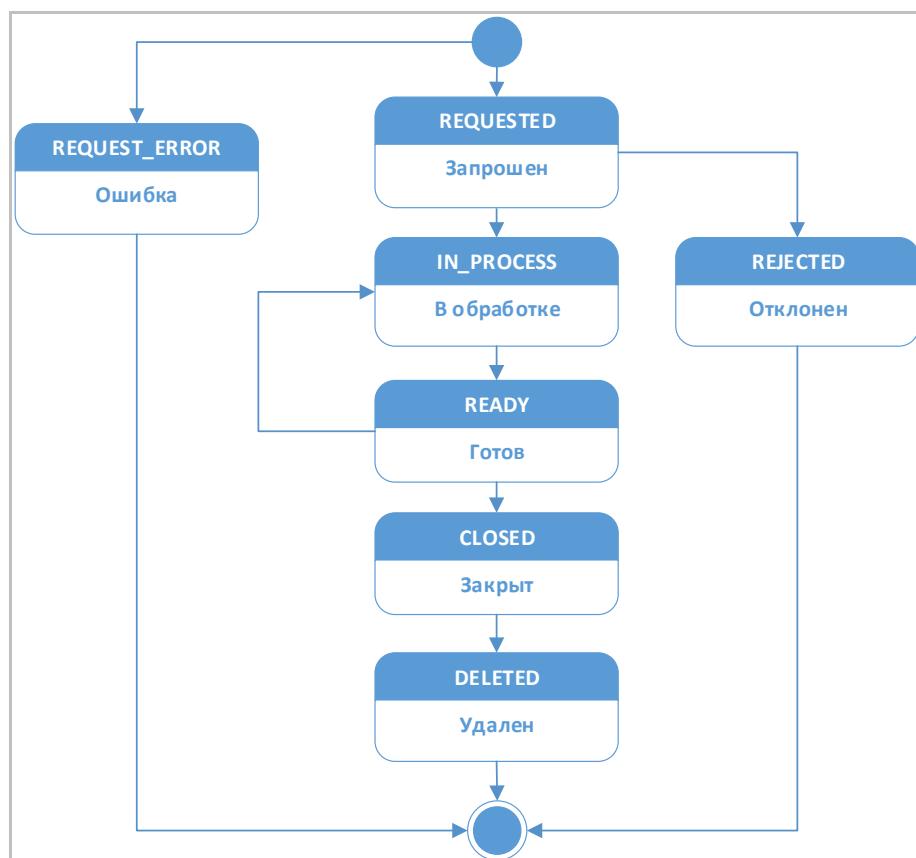
5.3.1.5. Catalog <MC Array Status>

For the list of possible values of Catalog <MC Array Status>, see Table 76.

Table 76 – Possible Values of Catalog <MC Array Status>

Constant	Value	Type
REQUEST_ERROR	Incorrect request format	String
REQUESTED	Array (pool) of the MCs has been requested	String
IN_PROCESS	Processing in progress	String
READY	Array (pool) of the MCs is ready for use	String
CLOSED	All MCs in the array were fully used	String
DELETED	MC array was exhausted and closed	String
REJECTED	Order not completed (incorrect order parameters—e.g., the order contains non-unique serial numbers)	String

The status chart is shown in Figure 54.



MC Array Status

Figure 54

5.3.1.6. Catalog <Aggregation Type>

For the list of possible values of catalog <Aggregation Type>, see Table 77.

Table 77 – Possible Values of <Aggregation Type> Catalog

Constant	Value	Type
AGGREGATION	New aggregation	String
UPDATE	Update of existing aggregation	String

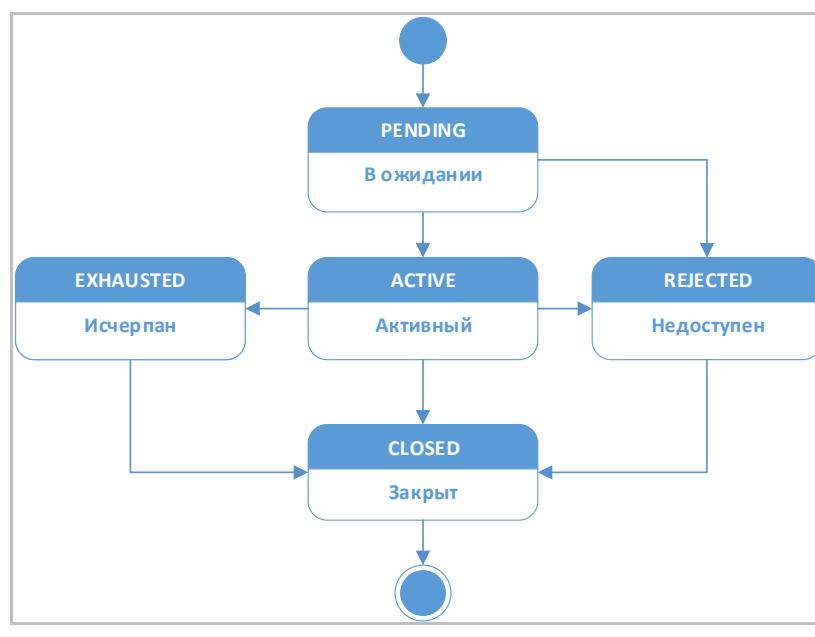
5.3.1.7. Catalog <MC Buffer Status>

For the list of possible values of Catalog <MC Buffer Status>, see Table 78.

Table 78 – Possible Values of Catalog <MC Buffer Status>

Constant	Value	Type
PENDING	MC buffer is pending	String
ACTIVE	Buffer is created	String
EXHAUSTED	No more codes left in the buffer and pools	String
REJECTED	Buffer is non-operational	String
CLOSED	Buffer is closed	String

The status chart is shown in Figure 55.



MC Buffer Status

Figure 55

5.3.1.8. Catalog <Report Processing Status>

For the list of possible values of Catalog <Report Processing Status>, see Table 79.

Table 79 – Possible Values of Catalog <Report Processing Status>

Constant	Value	Type
DRAFT	The report has been received by the OMS (it is deprecated and is not used)	String
PENDING	Report is pending	String
READY_TO_SEND	The report is ready for sending	String
REJECTED	Report is rejected	String
SENT	Report is sent	String

The status chart is shown in Figure 56.



Report Processing Status

Figure 56

5.3.1.9. Catalog <Type of Utilization>

For the list of possible values of Catalog <Type of Utilization>, see Table 80.

Table 80 – Possible Values of Catalog <Type of Utilization>

Constant	Value	Type
PRINTED	MC has been printed	String
VERIFIED	MC application—confirmed	String

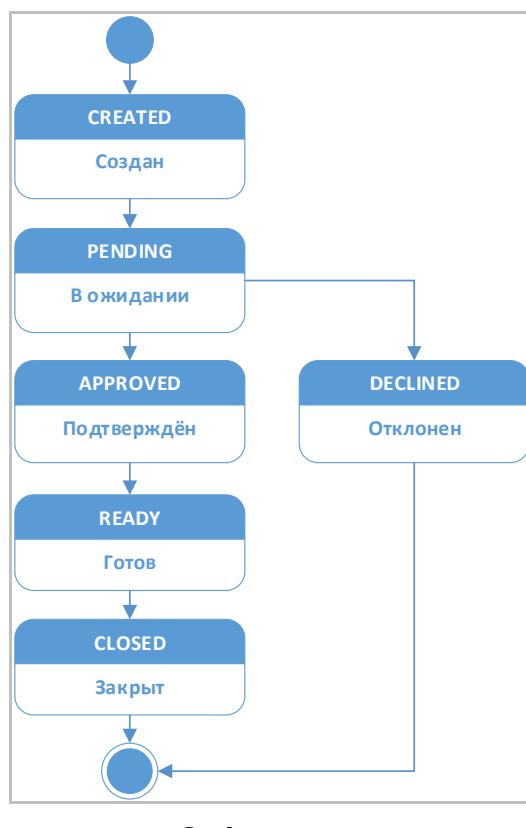
5.3.1.10. Catalog <Order status>

For the list of possible values of Catalog <Order status>, see Table 81.

Table 81 – Possible Values of Catalog <Order status>

Constant	Value	Type
CREATED	Order created	String
PENDING	The order is waiting for the confirmation in the marking system	String
DECLINED	Order is not confirmed in the marking system	String
APPROVED	The order is confirmed in the marking system	String
READY	Order ready	String
CLOSED	Order closed	String

The status chart is shown in Figure 57.



Order status

Figure 57

5.3.1.11. Catalog <Disposal Reason>

For the list of possible values of Catalog <Disposal Reason>, see Table 82.

Table 82 – Possible Values of Catalog <Disposal Reason>

Code	Constant	Description	Type
0	DEFECT	Defect	String
1	EXPIRY	Shelf life is expired	String
2	QA_SAMPLES	Laboratory test samples	String
3	PRODUCT_RECALL	Recalled from market	String
4	COMPLAINTS	Complaints	String
5	PRODUCT_TESTING	Product testing	String
6	DEMO_SAMPLES	Demo samples	String
7	OTHER	Other reasons	String

5.3.1.12. Catalog «Marking Code Type»

Table 83 shows a list of possible values for catalog «Marking Code Type».

Table 83 – Possible Values of Catalog «Marking Code Type»

Constant	Value	Type
"UNIT"	Goods item	String
GROUP	Group consumer packing	String

5.3.1.13. Catalog «Goods Groups»

For the list of possible values of catalog «Goods Groups», see Table 84.

Table 84 Possible Values of Catalog «Goods Groups»

Constant	Value	Type
shoes	Footwear	String
tobacco	Tobacco products	String
pharma	Medicines for human use	String
milk	Dairy products	String
alcohol	Alcohol	String
lp	Clothing items, bed, table, bath and kitchen linens	String
water	Packaged water and sugar-containing beverages	String

5.3.1.14. Catalog <International Classifier of Countries>

For the list of possible values of catalog <International classifier of countries>, see Table 85.

Table 85 – Possible Values of Catalog <International classifier of countries>

Constant	Value	Type
AU	AUSTRALIA	String
AT	AUSTRIA	String
AZ	AZERBAIJAN	String
AX	ALAND ISLANDS	String
AL	ALBANIA	String
DZ	ALGERIA	String
AS	AMERICAN SAMOA	String
AI	ANGUILLA (BRITISH)	String
AO	ANGOLA	String
AD	ANDORRA	String
AQ	ANTARCTICA	String
AG	ANTIGUA and BARBUDA	String
AR	ARGENTINA	String
AM	ARMENIA	String
AW	ARUBA	String
AF	AFGHANISTAN	String
BS	THE BAHAMAS	String
BD	BANGLADESH	String
BB	BARBADOS	String
BH	BAHRAIN	String
BY	BELARUS	String
BZ	BELIZE	String
BE	BELGIUM	String
BJ	BENIN	String
BM	BERMUDA	String
BG	BULGARIA	String
BO	BOLIVIA	String
BQ	Caribbean Netherlands	String
BA	BOSNIA and HERZEGOVINA	String
BW	BOTSWANA	String
BR	BRAZIL	String
IO	BRITISH INDIAN OCEAN TERRITORY	String
BN	BRUNEI	String
BV	BOUVET ISLAND	String
BF	BURKINA FASO	String

Constant	Value	Type
BI	BURUNDI	String
BT	BHUTAN	String
VU	VANUATU	String
GB	UNITED KINGDOM	String
HU	HUNGARY	String
VE	VENEZUELA	String
VG	BRITISH VIRGIN ISLANDS	String
VI	U.S. VIRGIN ISLANDS	String
VN	VIETNAM	String
GA	GABON	String
HT	HAITI	String
GY	GUYANA	String
GM	THE GAMBIA	String
GH	GHANA	String
GP	GUADELOUPE	String
GT	GUATEMALA	String
GN	GUINEA	String
GW	GUINEA-BISSAU	String
DE	GERMANY	String
GG	GUERNSEY	String
GI	GIBRALTAR (BRITISH)	String
HN	HONDURAS	String
HK	HONG KONG	String
GD	GRENADA	String
GL	GREENLAND	String
GR	GREECE	String
GE	GEORGIA	String
GU	GUAM (the U.S.)	String
DK	DENMARK	String
CD	DEMOCRATIC REPUBLIC OF THE CONGO	String
JE	JERSEY	String
DJ	DJIBOUTI	String
DM	DOMINICA	String
DO	DOMINICAN REPUBLIC	String
EU	EUROPEAN UNION	String
EG	EGYPT	String
ZM	ZAMBIA	String
EH	WESTERN SAHARA	String
ZW	ZIMBABWE	String
IL	ISRAEL	String
IN	INDIA	String

Constant	Value	Type
ID	INDONESIA	String
JO	JORDAN	String
IQ	IRAQ, THE REPUBLIC OF IRAQ	String
IR	IRAN, THE ISLAMIC REPUBLIC	String
IE	IRELAND	String
IS	ICELAND	String
ES	SPAIN	String
IT	ITALY	String
YE	YEMEN	String
CV	CAPE VERDE	String
KZ	KAZAKHSTAN	String
KH	CAMBODIA	String
CM	CAMEROON	String
CA	CANADA	String
QA	QATAR	String
KE	KENYA	String
CY	CYPRUS	String
KI	KIRIBATI	String
CN	CHINA	String
CC	COCOS (KEELING) ISLANDS	String
CO	COLOMBIA	String
KM	COMOROS	String
CG	Republic of the Congo	String
KP	THE DEMOCRATIC PEOPLE'S REPUBLIC OF KOREA	String
CR	COSTA RICA	String
CI	COTE D'IVOIRE	String
CU	CUBA	String
KW	KUWAIT	String
KG	KYRGYZSTAN	String
CW	CURAÇAO	String
LA	THE LAO PEOPLE'S DEMOCRATIC REPUBLIC	String
LS	LESOTHO	String
LR	LIBERIA	String
LB	LEBANON	String
LY	LIBYA	String
LT	LITHUANIA	String
LI	LIECHTENSTEIN	String
LU	LUXEMBOURG	String
MU	MAURITIUS	String
MR	MAURITANIA	String

Constant	Value	Type
MG	MADAGASCAR	String
YT	MAYOTTE	String
MO	MACAU	String
MK	MACEDONIA	String
MW	MALAWI	String
MY	MALAYSIA	String
ML	MALI	String
UM	UNITED STATES MINOR OUTLYING ISLANDS	String
MV	MALDIVES	String
MT	MALTA	String
MA	MOROCCO	String
MQ	MARTINIQUE	String
MH	MARSHALL ISLANDS	String
MX	MEXICO	String
FM	FEDERATED STATES OF MICRONESIA	String
MZ	MOZAMBIQUE	String
MD	THE REPUBLIC OF MOLDOVA	String
MC	MONACO	String
MN	MONGOLIA	String
MS	MONTSERRAT	String
MM	MYANMAR	String
NA	NAMIBIA	String
NR	NAURU	String
NP	NEPAL	String
NE	NIGER	String
NG	NIGERIA	String
NL	NETHERLANDS	String
NI	NICARAGUA	String
NU	NIUE	String
NZ	NEW ZEALAND	String
NC	NEW CALEDONIA	String
NO	NORWAY	String
AE	UNITED ARAB EMIRATES	String
OM	OMAN	String
IM	ISLE OF MAN	String
NF	NORFOLK ISLAND	String
CX	CHRISTMAS ISLAND	String
SH	SAINT HELENA ISLAND	String
HM	HEARD ISLAND AND MCDONALD ISLANDS	String
KY	CAYMAN ISLANDS	String
CK	COOK ISLANDS	String

Constant	Value	Type
TC	TURKS AND CAICOS ISLANDS	String
PK	PAKISTAN	String
PW	PALAU	String
PS	PALESTINIAN TERRITORIES, OCCUPIED	String
PA	PANAMA	String
VA	VATICAN CITY	String
PG	PAPUA NEW GUINEA	String
PY	PARAGUAY	String
PE	PERU	String
PN	PITCAIRN	String
PL	POLAND	String
PT	PORTUGAL	String
PR	PUERTO RICO	String
KR	REPUBLIC OF KOREA	String
LV	REPUBLIC OF LATVIA	String
RE	REUNION	String
RU	RUSSIA	String
RW	RWANDA	String
RO	ROMANIA	String
SM	SAN MARINO	String
WS	SAMOA	String
ST	SAO TOME AND PRINCIPE	String
SA	SAUDI ARABIA	String
SZ	SWAZILAND	String
VC	SAINT VINCENT AND THE GRENADINES	String
LC	SAINT LUCIA	String
MP	NORTHERN MARIANA ISLANDS	String
SC	SEYCHELLES	String
BL	SAINT BARTHELEMY	String
SN	SENEGAL	String
MF	SAINT MARTIN	String
SX	SAINT MARTIN (part of the Kingdom of the Netherlands)	String
PM	SAINT PIERRE AND MIQUELON	String
KN	SAINT KITTS AND NEVIS	String
RS	SERBIA	String
SG	SINGAPORE	String
SY	SYRIAN ARAB REPUBLIC	String
SK	SLOVAKIA	String
SI	SLOVENIA	String

Constant	Value	Type
US	UNITED STATES	String
SB	SOLOMON ISLANDS	String
SO	SOMALIA	String
SD	SUDAN	String
SR	SURINAME	String
SL	SIERRA LEONE	String
TJ	TAJIKISTAN	String
TH	THAILAND	String
TW	TAIWAN (CHINA)	String
TZ	UNITED REPUBLIC OF TANZANIA	String
TL	TIMOR-LESTE	String
TG	TOGO	String
TK	TOKELAU	String
TO	TONGA	String
TT	TRINIDAD AND TOBAGO	String
TV	TUVALU	String
TN	TUNISIA	String
TM	TURKMENISTAN	String
TR	TURKEY	String
UG	UGANDA	String
UZ	UZBEKISTAN	String
UA	UKRAINE	String
WF	WALLIS AND FUTUNA	String
UY	URUGUAY	String
FO	FAROE ISLANDS	String
FJ	FIJI	String
PH	PHILIPPINES	String
FI	FINLAND	String
FK	FALKLAND ISLANDS (MALVINAS)	String
FR	FRANCE	String
GF	FRENCH GUIANA	String
PF	FRENCH POLYNESIA	String
TF	FRENCH SOUTHERN TERRITORIES	String
HR	CROATIA	String
CF	THE CENTRAL AFRICAN REPUBLIC	String
TD	CHAD	String
ME	MONTENEGRO	String
CZ	THE CZECH REPUBLIC	String
CL	CHILE	String
CH	SWITZERLAND	String
SE	SWEDEN	String

Constant	Value	Type
SJ	SVALBARD AND JAN MAYEN	String
LK	SRI LANKA	String
EC	ECUADOR	String
GQ	EQUATORIAL GUINEA	String
SV	EL SALVADOR	String
ER	ERITREA	String
EE	ESTONIA	String
ET	ETHIOPIA	String
GS	SOUTH GEORGIA AND THE SOUTH SANDWICH ISLANDS	String
ZA	SOUTH AFRICA	String
JM	JAMAICA	String
JP	JAPAN	String

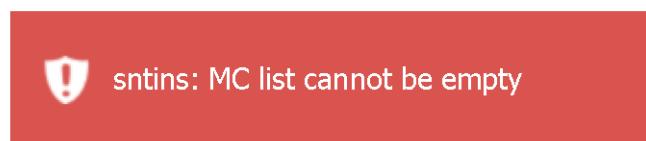
6. MESSAGES

6.1. Messages to Operator via GUI

6.1.1. Information windows

If any errors occur in the course of program execution, the web browser window will show a red pop-up message of one of the following two types:

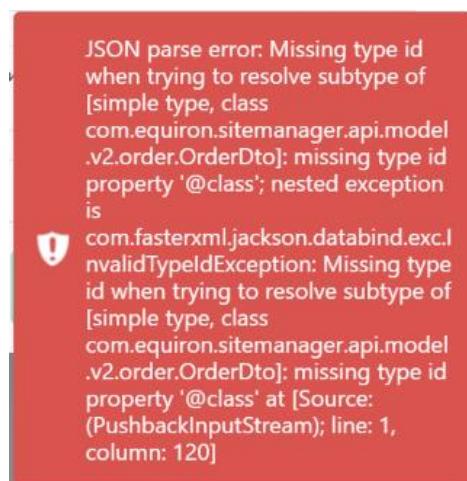
- 1) Program input error message (Figure 58).



Program input error message

Figure 58

- 2) System error message (Figure 59).



System error message

Figure 59

6.2. Error Format and Codes

6.2.1. Error format

For the format of response with an error, see Table 86.

Table 86 – Format of Response with Error

Field	Description	Type
fieldErrors	Error	JSON Array Of ProtobeanError Object
globalErrors	Description of global errors	JSON Array Of string
success	Request execution result	Boolean

Description of <ProtobeanError> object format is given in Table 87.

Table 87 – Format of <ProtobeanError> Object

Field	Description	Type
fieldError	Error description	String
fieldName	Field name	String

Example of JSON response with error is shown in Figure 60.

```
{
  "fieldErrors": [
    {
      "fieldError": "string",
      "fieldName": "string"
    }
  ],
  "globalErrors": [
    "string"
  ],
  "success": false
}
```

Example of JSON Response with Error

Figure 60

6.2.2. Description of Errors

For error codes in the request response, see Table 88.

Table 88 – Error Codes of Information Sending

Error code	Description
400	Operation not completed. Incorrect input parameters
500	Operation not completed. Internal server error

APPENDIX**APPLICATION PROGRAMMING INTERFACE REVISION HISTORY**

Version	Date	OMS version	List of Changes
11	24/11/2020	3.1.16	Support of HTTP header parameter - "X-Signature" was added
12	01/12/2020	3.1.16	<p>serviceProviderId, stickerId parameters were added in the method "Create marking code emission order" (see section 4.4.1) in order to support the marking code emission within the distribution.</p> <p>Method "Close the suborder by the GTIN specified" was renamed as "Close suborder/order" to support the closing of the whole order (see section 4.4.5), gtin parameter is optional in the request.</p> <p>Description of the structure was updated for the method "Get a status of MC array from the order" (see section 4.4.7).</p> <p>Description of the structure was updated for the method "Obtain the Report Processing Status" (see section 4.4.10).</p> <p>"Receive receipt by unique document identifier" method was added (see section 4.4.17).</p>
13	01/12/2020	3.1.16	Section on the description of HTTP header parameter - "X-Signature" was deleted due to disabling the features of signing of the documents (orders and reports)
14	05/02/2021	3.1.16	Support of the "Dairy products" goods group was added into the OMS API.
15	11/05/2021	3.1.16	<p>Order structure was updated for the "Dairy products" goods group (See section 4.4.1).</p> <p>Structure of the MC application reports was updated for the "Dairy products" goods group (See section 4.4.4).</p> <p>Structure of the MC aggregation reports was updated for the "Dairy products" goods group (See section 4.4.3).</p> <p>Structure of the MC rejection reports was updated for the "Dairy products" goods group (See section 4.4.2).</p>
17	21/06/2021	3.1.16.2	<p>Support of new optional parameter "country" was added into the order extension for the "Footwear", "Medicines for human use", "Dairy products", "Alcohol" goods groups (see sections 4.4.1.1.2, 4.4.1.1.3, 4.4.1.1.4, 4.4.1.1.5)</p> <p>Catalog "Method of Goods Release into Circulation" was expanded with new "REMAINS", "COMMISSION" values for "Footwear" goods group (see section 5.3.1.1)</p> <p>Support of new catalogs "Goods Groups" (see section 5.3.1.13) and "International classifier of countries" (see section 5.3.1.14) was added</p> <p>Note on the current marking code template for the "Medicines for human use" goods group was added into section 5.3.1.4</p>

Version	Date	OMS version	List of Changes
18	24/08/2021		<p>Support of “Clothing Items, Bed, Table, Bath and Kitchen Linen” and “Packaged water and sugar-containing beverages” goods groups was added, description of the order extension was indicated in the section for the goods group of the method “Create marking code emission order” (see section 4.4.1)</p> <p>Support of “Clothing Items, Bed, Table, Bath and Kitchen Linen” and “Packaged water and sugar-containing beverages” goods groups was added into the catalogs “MC Template” (see section 5.3.1.4) and “Goods Groups” (see section 5.3.1.13)</p> <p>Catalog “Method of Goods Release into Circulation” was expanded with new “REMARK” values for “Footwear” goods group (see section 5.3.1.1)</p> <p>Catalog “Method of Goods Release into Circulation” was expanded with new “REMARK” and “COMMISSION” values for “Clothing items, bed, table, bath and kitchen linen” goods group (see section 5.3.1.1)</p>

LIST OF TERMS

The following terms are used in this document:

- 1) Hardware means a PC (personal computer) or another computer equipment (mainframe, minicomputer, microcomputer, pocket PC, computer terminal).
- 2) Single-user hardware means the computing equipment that ensures:
 - automation of the computing component of daily activities of the Customer's employees;
 - access to the information services that provide automation for workflow across the Customer's company.
- 3) Shared hardware means the computing equipment intended for:
 - arrangement of computing platform that enables automation of the Customer's workflow;
 - monitoring and configuring of hardware that is a part of the automated system;
 - accumulation and processing of data used for automation of the Customer's workflow.
- 4) General software means a complex of program components providing minimal hardware functionality:
 - environment to launch and operate other software components (operating system);
 - tools to work with structured data sets (DBMS);
 - tools to access the Internet resources (web browser);
 - tools to publish hardware resources on the Internet (web server).
- 5) Specialized software means a complex of software components that are specifically developed for this particular hardware (not "packaged software").
- 6) End devices means the devices that are managed by the automated system directly (via the information and communication links) or indirectly (via the functional interface with the related systems), and are designed to perform technological functions (printer, scanner, controller, etc.).
- 7) Electronic Interface means aggregated resources and rules that enable interaction between the goods circulation participants' firmware and Track and Trace Information System.

- 8) Goods Code (GC, GTIN) is a unique code assigned to a group of goods while describing them at an information resource that enables accounting and storage of reliable data on the goods by corresponding codes of the unified Foreign Economic Activity Goods Nomenclature of the Eurasian Economic Union (the goods nomenclature).
- 9) Eurasian Economic Union (EEU) is an international organization of regional economic integration; it has the international legal personality and was established by the Treaty on the Eurasian Economic Union. EEU countries include: Republic of Armenia, Republic of Belarus, Republic of Kazakhstan, Kyrgyz Republic, and Russian Federation.

LIST OF ABBREVIATIONS

AWS	Automated Working station
AS	Automated system
APCS	Automated process control system
EEU	Eurasian Economic Union is an international organization of regional economic integration; it has the international legal personality and was established by the Treaty on the Eurasian Economic Union. EAEU countries include: Republic of Armenia, Republic of Belarus, Republic of Kazakhstan, Kyrgyz Republic, and Russian Federation
MC	Marking code
GC	Goods code, GTIN
HCS	Hardware Components Set
GSW	General Software
SW	Software
SHP	Software and Hardware Package
PC	Personal Computer
HW	Hardware
IM	Identification means
SSW	Special Software
OMS	Order Management Station
GCP	Goods circulation participant
SKU	Stock Keeping Unit

LIST OF PICTURES

Figure 1.....	15
Figure 2.....	17
Figure 3.....	19
Figure 4.....	21
Figure 5.....	23
Figure 6.....	25
Figure 7.....	31
Figure 8.....	32
Figure 9.....	34
Figure 10.....	36
Figure 11.....	38
Figure 12.....	40
Figure 13.....	42
Figure 14.....	43
Figure 15.....	45
Figure 16.....	46
Figure 17.....	47
Figure 18.....	48
Figure 19.....	53
Figure 20.....	54
Figure 21.....	55
Figure 22.....	57
Figure 23.....	58
Figure 24.....	59
Figure 25.....	60
Figure 26.....	61
Figure 27.....	63
Figure 28.....	64
Figure 29.....	66
Figure 30.....	67
Figure 31.....	68
Figure 32.....	70
Figure 33.....	71
Figure 34.....	73
Figure 35.....	74
Figure 36.....	76
Figure 37.....	77
Figure 38.....	78

Figure 39.....	79
Figure 40.....	80
Figure 41.....	81
Figure 42.....	82
Figure 43.....	83
Figure 44.....	84
Figure 45.....	85
Figure 46.....	86
Figure 47.....	87
Figure 48.....	88
Figure 49.....	90
Figure 50.....	92
Figure 51.....	93
Figure 52.....	94
Figure 53.....	96
Figure 54.....	103
Figure 55.....	105
Figure 56.....	106
Figure 57.....	108
Figure 58.....	118
Figure 59.....	118
Figure 60.....	119

LIST OF TABLES

Table 1 — List of the purpose indicators to which the system must conform.....	11
Table 2 – Valid MC Symbols.....	13
Table 3 – Request String Parameters.....	28
Table 4 – Description of JSON format of the request for creation and sending of the order for the MC emission, object <Order>	29
Table 5 – Format of <OrderProduct> Object	29
Table 6 – Structure of <Order> Object Extension for Tobacco Industry	31
Table 7 – Description of <Order> object extension for <Footwear> goods group	32
Table 8 – Description of <Order> object extension for <Alcohol> goods group category	33
Table 9 – Description of <OrderProduct> object extension for <Alcohol> goods group category ..	33
Table 10 – Structure of <Order> object extension for pharmacological industry	35
Table 11 – Structure of <Order> Object Extension for Dairy Industry.....	37
Table 12 – Description of <OrderProduct> Object Extension for <Dairy products> Goods Group Category	37
Table 13 – Structure of <Order> Object Extension for “Items of Clothing, Bed, Table, Bath and Kitchen Linen” Goods Group	39
Table 14 – Description of <OrderProduct> Object Extension for <Items of Clothing, Bed, Table, Bath and Kitchen Linen> Goods Group	39
Table 15 – Structure of <Order> Object Extension for <Packaged water and sugar-containing beverages> Goods Group	41
Table 16 – Description of <OrderProduct> Object Extension for <Packaged water and sugar- containing beverages> Goods Group	41
Table 17 – Format of Response to Request	43
Table 18 – Request String Parameters.....	44
Table 19 – <DropoutReport> Object Structure	44
Table 20 – Description of <DropoutReport> Object Extension for Tobacco Industry.....	45
Table 21 – Description of <DropoutReport> Object Extension for Pharmaceutical Industry.....	46
Table 22 – Description of <DropoutReport> Object Extension for Dairy Industry	47
Table 23 – Format of Response to Request to Send Report on MC Dropout / Rejection	48
Table 24 – Request String Parameters.....	49
Table 25 – Structure of <AggregationReport> Object.....	49
Table 26 – Structure of <AggregationUnit> Object	50
Table 27 – Description of “AggregationReport” Object Extension for Tobacco Industry	53
Table 28 – Description of “AggregationReport” Object Extension for Pharmaceutical Industry .	54
Table 29 – Format of Response to Request to Send Information on Aggregation.....	55
Table 30 – Request String Parameters.....	56
Table 31 – Structure of “UtilisationReport” Object	56
Table 32 – Description of <UtilisationReport> Object Extension for Tobacco Industry.....	57

Table 33 – Description of <UtilisationReport> Object Extension for Pharmaceutical Industry	58
Table 34 – Description of <UtilisationReport> Object Extension for Dairy Industry	59
Table 35 – Description of <UtilisationReport> Object Extension for <Packaged water and sugar-containing beverages> goods group.....	60
Table 36 – Format of Response to Request to Send Report on MC Application	61
Table 37 – Request String Parameters.....	62
Table 38 – Format of Response to Request to Close Suborder by GTIN Specified	64
Table 39 – Request String Parameters.....	65
Table 40 – Format of Response to Request for MC for the Goods Specified	67
Table 41 – Request String Parameters.....	68
Table 42 – Format of Response to Request, object “BufferInfo”	69
Table 43 – <PoolInfo> Object Format.....	70
Table 44 – Request String Parameters.....	71
Table 45 – Format of Response to Request for Order Status.....	72
Table 46 – <OrderSummaryInfo> Object Format	72
Table 47 – Request String Parameters.....	74
Table 48 – Format of Response to Request for Aggregation Information, object <AggregationInfo>	75
Table 49 – <ProductInfo> Object Format.....	75
Table 50 – Description of <AggregationInfo> Object Extension for Manufacturers of Tobacco Industry	77
Table 51 – Description of <AggregationInfo> Object Extension for Manufacturers of Pharmaceutical Industry	78
Table 52 – Request String Parameters.....	79
Table 53 – Format of Response to Request for Report Processing Status.....	80
Table 54 – Request String Parameters.....	81
Table 55 – Request Body Parameters.....	81
Table 56 – Format of Response to Request for OMS Availability	82
Table 57 – Request String Parameters.....	83
Table 58 – Format of Response to Request for OMS Availability	84
Table 59 – HTTP Header Parameters	85
Table 60 – Request String Parameters.....	85
Table 61 – Format of Response to Request for Security Marker	86
Table 62 – Format of Response to Request for OMS and API Version	87
Table 63 – Request String Parameters.....	88
Table 64 – Format of Response to Request for the List of MC Packet Identifiers for the Marking and Goods Code Order Specified.....	89
Table 65 – Format of MC Packet List, Object <Block>	89
Table 66 – Request String Parameters.....	91
Table 67 – Format of Response to Request for Re-obtainment of MC for the Goods Specified	93

Table 68 – Request String Parameters.....	94
Table 69 – Format of Response to Request	95
Table 70 – Format of Receipt Object	95
Table 71 – Possible Values of Catalog «Method of Goods Release into Circulation».....	100
Table 72 – Possible Values of Catalog «Method of Individual Serial Number Generation»	100
Table 73 – Possible Values of Catalog «Manufacturing method»	100
Table 74 – Possible Values of Catalog «MC Template»	101
Table 75 – Description of MC Templates.....	101
Table 76 – Possible Values of Catalog «MC Array Status».....	103
Table 77 – Possible Values of «Aggregation Type» Catalog.....	104
Table 78 – Possible Values of Catalog «MC Buffer Status».....	105
Table 79 – Possible Values of Catalog «Report Processing Status»	106
Table 80 – Possible Values of Catalog «Type of Utilization»	107
Table 81 – Possible Values of Catalog «Order status»	108
Table 82 – Possible Values of Catalog «Disposal Reason».....	109
Table 83 – Possible Values of Catalog «Marking Code Type».....	110
Table 84 Possible Values of Catalog «Goods Groups»	110
Table 85 – Possible Values of Catalog «International classifier of countries».....	111
Table 86 – Format of Response with Error	119
Table 87 – Format of «ProtobeanError» Object.....	119
Table 88 – Error Codes of Information Sending.....	120

132

KZ 15861920.620111-04 33 01