



NAV e-Cash register system

e-Cash Register Developer Documentation



1	TERMS, ABBREVIATIONS.....	7
2	DOCUMENT HISTORY	8
3	INTRODUCTION	11
3.1	PURPOSE	11
3.2	DEVELOPMENT PATHS FOR THE END PRODUCTS.....	12
3.2.1	<i>Hardware-Based E-Cash Register</i>	12
3.2.2	<i>Cloud-Based E-Cash Register Application</i>	12
3.2.3	<i>Customer Application.....</i>	13
3.3	SUBMITTING A CONNECTION REQUEST	13
3.4	USAGE REQUIREMENTS FOR TAXPAYERS.....	14
3.5	TECHNOLOGIES TO BE IMPLEMENTED FOR CONNECTION	14
3.6	TECHNICAL REQUIREMENTS FOR E-CASH REGISTER SOFTWARE.....	15
3.7	TECHNICAL REQUIREMENTS FOR CUSTOMER APPLICATIONS	15
3.8	CERTAIN TECHNICAL CONDITIONS FOR THE AUTHORIZATION OF MOBILE APPLICATIONS	15
4	GENERAL DESCRIPTION OF THE E-RECEIPT MACHINE INTERFACE	16
4.1	PROCESS OF eRECEIPT DATA SERVICES	16
4.2	CREATING SIGNATURES	16
4.3	AUTHENTICATION	17
4.4	COMPRESSION	18
4.5	ENCRYPTION	19
4.5.1	<i>Generation of encryption key pair by the E-Cash register.....</i>	19
4.5.2	<i>Data encryption.....</i>	19
4.6	GENERATING QR CODES AND NDEF (NFC) DATA PACKAGES BY THE E-CASH REGISTER	21
4.6.1	<i>Generating input QR code by the E-Cash register.....</i>	21
4.6.2	<i>Formation of the output QR code for the E-Cash register</i>	29
4.6.3	<i>Generation of the E-Cash register signature verification QR code</i>	32
4.6.4	<i>Interpretation of Cloud-based e-Cash Register Activation QR Code.....</i>	36
4.6.5	<i>Creating an NDEF record for input code confirmation.....</i>	37
4.6.6	<i>NDEF record creation for requesting a receipt envelope</i>	38
4.6.7	<i>NDEF record creation for receipt envelope.....</i>	39
4.7	NAV VERIFICATION CODE GENERATION	41
4.7.1	<i>Example of NAV verification code calculation.....</i>	42
4.8	TECHNICAL DESCRIPTION OF THE SERVICES	42
4.8.1	<i>General technical data</i>	42
4.8.2	<i>HTTP headers</i>	49
4.8.3	<i>HTTP status codes.....</i>	49
4.8.4	<i>Response time, timeout</i>	49
4.8.5	<i>Converting local time to UTC</i>	49
4.8.6	<i>Radius server.....</i>	50
4.9	RECEIPT ISSUANCE, SUBMISSION, AND QUERYING TECHNOLOGICAL PROCESS.....	50
4.9.1	<i>When using a customer application</i>	50
4.9.2	<i>Without using a customer application.....</i>	51
4.10	SEARCH KEY	52
5	BUSINESS SERVICES PROVIDED BY NAV FOR E-CASH REGISTERS.....	52
5.1	DEVICE REGISTRATION.....	52
5.1.1	<i>Business description of the service.....</i>	52
5.1.2	<i>Technical description of the service</i>	53
5.2	RECEIPT SUBMISSION	55
5.2.1	<i>Business description of the service.....</i>	56
5.2.2	<i>Receipt envelope compilation.....</i>	57
5.2.3	<i>Technical description of the service</i>	59
5.3	REPORT RECEPTION.....	62
5.3.1	<i>Business description of the service.....</i>	62
5.3.2	<i>Compilation of the report document envelope</i>	63
5.3.3	<i>Technical description of the service</i>	65
5.4	E-CASH REGISTER STATUS REPORT	68



5.4.1	<i>Business description of the service</i>	68
5.4.2	<i>Technical description of the service</i>	69
5.5	COMMUNICATION MANAGER.....	70
5.5.1	<i>Business description of the service</i>	70
5.5.2	<i>Technical description of the service</i>	71
5.6	TAXPAYER DATA QUERY	73
5.6.1	<i>Business description of the service</i>	73
5.6.2	<i>Technical description of the service</i>	74
5.7	VAT MASTER DATA QUERY	75
5.7.1	<i>Business description of the service</i>	75
5.7.2	<i>Technical description of the service</i>	76
5.8	E-CASH REGISTER BLOCKING/UNBLOCKING	78
5.8.1	<i>Business description of the service</i>	78
5.8.2	<i>Technical description of the service</i>	79
5.9	SENDING TECHNICAL INFORMATION	80
5.9.1	<i>Business description of the service</i>	80
5.9.2	<i>Technical description of the service</i>	81
5.10	SOFTWARE UPDATE.....	82
5.10.1	<i>Business description of the service</i>	82
5.10.2	<i>Technical description of the service</i>	82
5.11	DOMESTIC TAX NUMBER VERIFICATION	83
5.11.1	<i>Business description of the service</i>	83
5.11.2	<i>Technical description of the service</i>	83
5.12	TERMINATION OF OPERATION	85
5.12.1	<i>Business description of the service</i>	85
5.12.2	<i>Technical description of the service</i>	86
5.13	RESUMPTION OF OPERATION	88
5.13.1	<i>Business description of the service</i>	88
5.13.2	<i>Technical description of the service</i>	88
5.14	RE-PERSONALIZATION	88
5.14.1	<i>Business description of the service</i>	88
5.14.2	<i>Technical description of the service</i>	89
5.15	HELLO SERVICE.....	93
5.15.1	<i>Business description of the service</i>	93
5.15.2	<i>Technical description of the service</i>	93
5.16	CERTIFICATE RENEWAL.....	95
5.16.1	<i>Business description of the service</i>	95
5.16.2	<i>Technical description of the service</i>	95
5.17	PRODUCT CATALOG QUERY.....	98
5.17.1	<i>Business description of the service</i>	98
5.17.2	<i>Technical description of the service</i>	98
5.18	REQUEST FOR SUBMISSION OF MISSING DOCUMENTS.....	102
5.18.1	<i>Business description of the service</i>	102
5.18.2	<i>Technical description of the service</i>	102
6	BUSINESS SERVICES PROVIDED BY MOBILE SERVICE PROVIDERS/CLOUD SERVICE PROVIDERS	104
6.1	INSTRUCTION FOR IMMEDIATE LOGIN.....	104
7	CUSTOMER APPLICATION REQUIREMENTS.....	104
7.1	CUSTOMER APPLICATION REGISTRATION	105
7.2	KEY GENERATION.....	105
7.3	RECEIPT DOWNLOAD	106
7.4	RECEIPT DECRYPTION.....	106
7.5	EXPORT.....	107
7.6	SAVING RECEIPT HEADER DATA FOR RECOVERY FROM THE RECEIPT REPOSITORY	107
7.7	IMPORT	109
7.8	RECOVERY	109
7.9	QR CODE GENERATION	109



7.10	VERSION CONTROL.....	109
7.11	HANDLING OF INSPECTION AND LIVE MODES.....	109
8	SERVICES PROVIDED BY THE RECEIPT REPOSITORY	110
8.1	TECHNICAL DESCRIPTION OF SERVICES	110
8.1.1	<i>General Technical Data</i>	110
8.1.2	<i>HTTP headers</i>	110
8.1.3	<i>HTTP status codes</i>	111
8.1.4	<i>Response time, timeout</i>	111
8.2	CUSTOMER APPLICATION REGISTRATION	112
8.2.1	<i>Business description of the service</i>	112
8.2.2	<i>Technical description of the service</i>	112
8.3	RECEIPT QUERY	115
8.3.1	<i>Business description of service</i>	115
8.3.2	<i>Technical description of the service</i>	115
8.4	SUBMISSION OF EVENTS OCCURRING IN THE CUSTOMER APPLICATION	119
8.4.1	<i>Business description of the service</i>	119
8.4.2	<i>Technical description of the service</i>	120
8.5	RETRIEVING NOTIFICATION DATA	122
8.5.1	<i>Business description of the service</i>	122
8.5.2	<i>Technical description of the service</i>	122
8.6	RETRIEVING NOTIFICATION DOWNLOAD WORKFLOWS	124
8.6.1	<i>Business description of the service</i>	124
8.6.2	<i>Technical description of the service</i>	124
9	STORAGE OF RECEIPTS ON HARDWARE-BASED E-CASH REGISTERS.....	126
10	DESCRIPTION OF BUSINESS DATA CONTENT (XSD MODEL TYPES AND ELEMENTS)..	127
10.1	ERECEIPTAPI.XSD	127
10.1.1	<i>XSD Element list</i>	127
10.2	DOCUMENTMESSAGE.XSD.....	129
10.2.1	<i>XSD Element list</i>	129
10.3	REPORTMESSAGE.XSD.....	129
10.3.1	<i>XSD Element lista</i>	129
10.4	DOCUMENTDATA.XSD.....	130
10.4.1	<i>XSD Simple type lista</i>	130
10.5	COMMUNICATIONDATA.XSD	131
10.5.1	<i>XSD Simple type list</i>	131
10.6	ERECEIPTBASE.XSD	137
10.6.1	<i>XSD Simple type list</i>	137
10.7	EDOCUMENTSTOREAPI.XSD.....	145
10.7.1	<i>XSD Element list</i>	145
10.8	EDOCUMENTSTOREMESSAGE.XSD.....	145
10.8.1	<i>XSD Simple type list</i>	145
10.9	ERECEIPTEXPORT.XSD	145
10.9.1	<i>XSD Element list</i>	145
11	ERROR HANDLING	146
11.1	ERROR RESPONSE.....	146
11.2	TECHNICAL ERROR CODES.....	146
11.2.1	<i>Common error codes</i>	146
11.2.2	<i>ECash register interface error codes</i>	148
11.2.3	<i>Receipt repository response codes</i>	153
11.3	VALIDATION ERRORS AND ERROR CODES	153
12	CLOUD-BASED FISCAL MODULE (FAM).....	155
12.1	REGISTRATION SERVICE	156
12.2	E-CASH REGISTER REST INTERFACE – GENERAL INFORMATION	157
12.2.1	<i>General request validations and response messages</i>	159
12.2.2	<i>Key Data Fields (and their value sets)</i>	161



12.2.3	<i>Key object descriptions</i>	163
12.3	USER AUTHENTICATION	165
12.3.1	<i>Requesting a client authentication certificate</i>	166
12.3.2	<i>Client authentication certificate renewal</i>	167
12.3.3	<i>Client authentication certificate download</i>	168
12.3.4	<i>Login</i>	169
12.3.5	<i>Logout</i>	171
12.4	STATE MANAGEMENT.....	171
12.4.1	<i>Querying the FAM instance status</i>	172
12.4.2	<i>Querying the FAM system status</i>	178
12.4.3	<i>Document management status</i>	182
12.5	TELEMETRY	183
12.5.1	<i>Responses, error codes</i>	183
12.5.2	<i>Hello</i>	183
12.5.3	<i>Domestic Tax Number query</i>	184
12.5.4	<i>Sending cash register state information</i>	186
12.5.5	<i>Sending cash register information</i>	187
12.6	CURRENCY MANAGEMENT	188
12.6.1	<i>Key object descriptions and data fields</i>	188
12.6.2	<i>Querying a specific currency type</i>	188
12.6.3	<i>Querying all currencies</i>	189
12.6.4	<i>Deleting a currency</i>	190
12.6.5	<i>Adding or modifying a currency type</i>	191
12.7	PAYMENT METHODS.....	191
12.7.1	<i>Key object descriptions and data fields</i>	192
12.7.2	<i>Querying a payment method</i>	192
12.7.3	<i>Retrieving all payment methods</i>	193
12.7.4	<i>Deleting a payment method</i>	194
12.7.5	<i>Adding or modifying a payment method</i>	194
12.7.6	<i>Querying all predefined payment methods</i>	195
12.8	PERIPHERAL MANAGEMENT	198
12.8.1	<i>Peripheral settings</i>	198
12.8.2	<i>Querying settings</i>	198
12.8.3	<i>Storing a setting</i>	199
12.9	RECEIPT MANAGEMENT	200
12.9.1	<i>Automatic end-of-day closure</i>	201
12.9.2	<i>Response messages and error codes</i>	201
12.9.3	<i>Key object descriptors and data fields</i>	203
12.9.4	<i>Cash Register opening receipt</i>	219
12.9.5	<i>Receipt</i>	221
12.9.6	<i>Cancellation document</i>	228
12.9.7	<i>Money movement receipt</i>	235
12.9.8	<i>Cash register report</i>	238
12.9.9	<i>Daily sales report</i>	240
12.9.10	<i>Modification receipt</i>	242
12.9.11	<i>Receipt summary report</i>	250
12.9.12	<i>Simplified invoice</i>	251
12.9.13	<i>Invoice</i>	257
12.9.14	<i>Custom document</i>	264
12.9.15	<i>Data transfer from customer application</i>	268
12.9.16	<i>Modification of the open receipt type</i>	269
12.9.17	<i>Verification of payment methods</i>	270
12.10	QUERYING THE DOCUMENT LIST BASED ON A TIME INTERVAL, OPTIONALLY BY TYPE	272
12.11	RETRIEVING THE DATA STRUCTURE USED FOR RECEIPT IMAGE GENERATION	273
12.12	REQUESTING A COMPLETE RECEIPT IMAGE	277
12.13	FILE MANAGEMENT.....	278
12.14	AUDITOR IDENTIFICATION.....	279
13	OPTIONAL SUPPLEMENTARY FIELDS FOR DOCUMENTS.....	280



13.1	THE PURPOSE OF THE SUPPLEMENTARY INFORMATION ELEMENT	280
13.2	SALES RECEIPTS	280
13.3	REPORTS	281
13.4	STRUCTURE OF THE SUPPLEMENTARY INFORMATION ELEMENT	281
13.5	EXAMPLES TO HELP INTERPRETATION	282
13.5.1	<i>Case A</i>	282
13.5.2	<i>Case B</i>	282
13.5.3	<i>Case C</i>	283
13.5.4	<i>Case D</i>	283
13.6	LIST OF STANDARD SUPPLEMENTARY ELEMENTS	284
13.7	DOCUMENT ATTACHMENT	285
13.8	USE CASES	285
13.8.1	<i>Warranty information</i>	286
13.8.2	<i>Coupon information</i>	288
13.8.3	<i>Quick access barcode</i>	289
13.8.4	<i>Line – or dot code</i>	290
13.8.5	<i>Image</i>	290
13.8.6	<i>General file attachment reference</i>	291
14	DATA TRANSFERS BETWEEN CUSTOMER APP AND E-CASH REGISTER	293
14.1	DATA TRANSFER FROM THE CUSTOMER APPLICATION FOR AN OPEN RECEIPT	294
14.1.1	<i>QR Code data transfer process</i>	295
14.1.2	<i>Data transfer via NFC</i>	296
14.2	DATA TRANSMISSION AT RECEIPT CLOSURE	297
14.2.1	<i>Transmission via QR code or NFC</i>	298
14.2.2	<i>Retrieving the receipt envelope via NFC</i>	298
15	OTHER RESTRICTIONS ON THE OPERATION OF E-CASH REGISTERS	300
15.1	PRINTING	300
15.2	RELATIONSHIP BETWEEN FISCAL DAY AND CALENDAR DAY	300
15.3	IN-STORE CASH WITHDRAWAL (CASH BACK)	300
15.4	DISPLAYING VAT GROUP INFORMATION	301
16	MASTER DATA	302
16.1	COMPETENCE CODES INDICATING THE RELEVANT STATE TAX AUTHORITY (COUNTYCODE)	302
16.2	COUNTRY CODE TYPE ACCORDING TO ISO 3166 ALPHA-2 STANDARD	303
16.3	POSTAL CODE DATABASE AVAILABILITY	303
16.4	VTSZ (CUSTOMS TARIFF NUMBER) DATABASE AVAILABILITY	303
16.5	SZJ (SERVICE CLASSIFICATION NUMBER) DATABASE AVAILABILITY	303
16.6	KN (COMBINED NOMENCLATURE) DATABASE AVAILABILITY	303
16.7	CSK (PACKAGING MATERIAL CATALOGUE CODE) DATABASE AVAILABILITY	303
16.8	KT (ENVIRONMENTAL PRODUCT CODE) DATABASE AVAILABILITY	303
16.9	EJ (BUILDING REGISTER NUMBER) DATABASE AVAILABILITY	303
16.10	TESZOR (PRODUCT AND SERVICE CLASSIFICATION SYSTEM) DATABASE AVAILABILITY	303
17	ACCESSIBILITY OF THE ENVIRONMENTS	304
17.1	INSPECTION ENVIRONMENT	304
17.1.1	<i>URLs of the inspection environment</i>	304
17.1.2	<i>Development support resources</i>	304
17.2	PRODUCTION ENVIRONMENT	305
17.2.1	<i>URLs of the production environment</i>	305
18	HELPDESK AND TECHNICAL SUPPORT	305
18.1	HELPDESK AVAILABILITY	305
19	VERSION TRACKING	305
19.1	VERSION 1.0	307
19.2	VERSION 1.1	307



1 Terms, abbreviations

Term	Description
Taxpayer	A taxpayer registered in Hungary who operates a cash register based on the relevant legislation.
Tax Unit (AE)	A separate, closed, electronic data storage and mobile communication unit in hardware-based e-cash registers, which creates, stores, and ensures data specified in the VAT law and the cash register regulations, along with maintaining encrypted communication between the e-cash register and NAV-I.
Cloud-Based Tax Module (FAM)	A virtual tax unit service mandatorily used by cloud-based e-cash registers.
Signing Key	A unique key that ensures the authenticity of the data signed with it.
API	Application Programming Interface.
VAT Act.	Act CXXVII of 2007 on Value Added Tax.
Control Code	A hexadecimal value generated by an algorithm using the control code of the previous receipt and the data content of the current receipt to verify the integrity of the recorded data content.
Endpoint	An access point through which the services provided by the operation can be reached.
ePG Portal	See KOBÁK Portal
Process Identifier	A unique identifier provided by NAV-I to track the status of processes initiated on e-cash registers, such as taxpayer data updates, VAT updates, software updates, blocking, or re-sending receipts.
Hardware Fingerprint	An identifier sequence that determines, with high certainty, the hardware device on which specific software is installed. This sequence does not identify the hardware user under any circumstances.
AE Manufacturer	The organization that manufactures the tax unit (AE) integrated into the e-cash register, storing tax-related information and transmitting it to NAV-I. If the AE is manufactured outside Hungary, the manufacturer's Hungarian representative with legal personality qualifies as the AE manufacturer. In the absence of representation, the manufacturer may designate the distributor of the e-cash register for representation via a notarized authorization.
IMEI	A unique identifier assigned to GSM devices.
IMSI	International Mobile Subscriber Identifier, allowing unique identification of every subscriber in a unified international system.
JSON	JSON (JavaScript Object Notation), an open standard file format for data exchange.
Search Key	A unique identifier for the receipt, necessary for querying it from the receipt database.



Term	Description
Display	A component of the e-cash register for displaying relevant information to the operator and/or the customer. In cloud-based e-cash registers, the mobile device running the client application acts as the display.
KOBAK Porta	A portal interface for cash register distributors and operators where distributors can set up and test e-cash registers. It also facilitates the submission of licensing requests.
Opening Receipt	See Cash Register Opening Receipt.
NAV (NTCA)	National Tax and Customs Administration of Hungary.
Operation	IT procedures or services are callable through the offered web service.
Cash Register	In this document, a cash register refers to an e-cash register as defined by the 8/2025. (III.31) NGM decree unless otherwise specified.
Cash Register Regulation	the 8/2025. (III.31) NGM decree on the distribution, operation, and requirements for e-cash registers and e-receipts.
Cash Register Opening Receipt	A type of receipt that records the initiation of the Tax Day (“opening receipt”).
QR code	A point code according to ISO/IEC 18004 standard.
SHA-256	A 256-bit Secure Hash Algorithm (RFC6234).
SHA3-512	A 512-bit Secure Hash Algorithm (Secure Hash Algorithm 3, RFC6931).
Parent Element	An element in a schema file that contains other elements.
Activation	The initial start of an e-cash register intended for end users or its operational use after a change in the operator's identity.
XML	Extensible Markup Language (eXtensible Markup Language, W3C standard https://www.w3.org/TR/xml/).
XSD	XML Schema Definition file (XML Schema Definition, W3C standard https://www.w3.org/TR/xmlschema11-1/).
Web service	A collection of protocols and standards for data exchange between applications.

2 Document history

Date	Author	Version	Change
2025.09.19	NAV	1.4	Updated version (containing all previous changes) 6.1 Deleted immediate login required FePG push message 8.4.1 New code added to Customer application 10. New data model for XSD



Date	Author	Version	Change
			11. Error handling numbering change 13.6 New appendix- General 13.8.6 Description of the new appendix 19.2 New start date of the mandatory use of interface v 1.1
2025.08.01	NAV	1.3	3.8 New subsection on certain authorization conditions for mobile applications 4.5.2: Clarification of figure and description (encryption) 4.6.2: Supplement to the output QR code description 7.3: Supplement to the description of customer app receipt download 7.10: New subsection on customer application version control 7.11: New subsection on customer application environment connection 8: Clarification of context roots in the subsections 10.11.3: New receipt store error code 11.2.1: Clarification of FAM request validation 11.4.1.2: Clarification of description 11.4.2.1: New FAM status query endpoint 11.9.2: Supplement to FAM document interface response codes 11.9.3.1.4: Supplement to FAM customInfo data structure Modification of customInfo examples in several FAM subsections in accordance with 11.9.3.1.4 11.9.3.2: Clarification of FAM receipt attachment value field length 11.9.3.5: Extension of FAM billTo data structure with the group member's tax number 11.9.10.2: Addition of FAM receipt item, extension of item type list 12: Clarification of supplementary information element description in several subsections, extension of standard elements, new use cases 18: Clarification of version management, introduction of version v1.1
2025.06.10	NAV	1.2	11.9.3.2: Modification of FAM receipt-attachment data structure



Date	Author	Version	Change
2025.05.23	NAV	1.1	4.5.2: Clarification of encryption 4.6.4: Extension of ePG output QR code interpretation 5.11: Clarification of domestic tax number query service 7, 7.1: Clarification of requirements for the customer application 8.2: Clarification of customer application registration service 8.4.2: Clarification of customer application event submission objects 8.5: Clarification of customer notifications download service 8.6: Retrieval of customer notification download workflows (new) 11.1: Explanation to assist the choice of endpoint URLs 12.8.1: Correction of Example 1 16.1: Clarification of URL list
2025.04.03	NAV	Draft 1.0	Updated draft
2025.03.21	NAV	Draft 1.0	Updated draft
2025.01.17	NAV	Draft 1.0	Updated draft
2023.10.02	NAV	Draft 1.0	Draft



3 Introduction

3.1 Purpose

The purpose of this document is to describe the operation of services to be used by e-cash registers and customer applications, as well as the XML message structures utilized by these services.

This document encompasses the business and technical content of the following schema descriptors.

Schema	Content
common.xsd	Generic types, catalog elements, and primitives describing NAV communication.
eReceiptBase.xsd	System-specific data types for the e-cash register (excluding invoices and simplified invoices).
invoiceBase.xsd	Specific data types for invoices and simplified invoices.
invoiceData.xsd	Business content of the invoice and simplified invoice services.
documentData.xsd	Business content of the receipt submission service (excluding invoices and simplified invoices).
communicationData.xsd	Business content of communication cases within the e-cash register system.
documentMessage.xsd	Elements of envelopes containing receipt parts for the receipt submission operation.
reportMessage.xsd	Elements of envelopes used in the report submission operation.
eReceiptApi.xsd	Request and response structures of services callable by e-cash registers.
eDocumentStoreApi.xsd	Request and response structures of services provided by the receipt store to customer applications.
eDocumentStoreMessage.xsd	Specific data types for services provided by the receipt store to customer applications.
eReceiptExport.xsd	XML export structure for customer applications.

The general rules for the services are detailed in the chapter „[General Description of the eReceipt machine interface](#)” while the description of services to be utilized by e-cash registers can be found in the chapter “[Business Services Provided by NAV for E-Cash Registers.](#)” The elements of the XSD schemas are described in the chapter “[Description of business data content \(XSD model types and elements\)](#)”

Additionally, the document includes the chapter “[Business services provided by mobile/cloud service providers](#)” through which NAV-I can issue instructions to individual e-cash registers. The chapter “[Customer Application Requirements](#)” outlines the requirements to be met by customer applications designed to query receipts. The customer application can query receipts



from the receipt store using the services described in the chapter "[Services provided by the Receipt Repository](#)".

3.2 Development Paths for the end products

The e-receipt ecosystem defines three main end products:

1. Hardware-based e-cash register
2. Cloud-based e-cash register application
3. Customer application

Different chapters of this document provide support for the development of each of these end products.

3.2.1 Hardware-Based E-Cash Register

The development of the hardware-based e-cash register (HePG) includes the following activities:

- Developing a Tax Unit (AE) hardware module that meets the requirements specified in the regulation.
- Implementing the software requirements for the Tax Unit.
 - Implementing the operational logic and processes described in the chapter "[General Description of the eReceipt machine interface](#)"
 - Realizing the communication protocol between the Tax Unit and NAV-I according to the requirements detailed in the chapter "[Business Services Provided by NAV for E-Cash registers](#)".
 - Structuring the communication data between the Tax Unit and NAV-I by applying the data structures described in the chapter "[Description of business data content \(XSD model types and elements\)](#)"
 - Implementing the Tax Unit's components of the services described in the chapter "[Business services provided by mobile/cloud service providers](#)"
 - Developing data storage capabilities for the data generated in the Tax Unit according to the chapter "[Storage of receipts on hardware-based E-Cash Registers](#)".
- Preparing the e-cash register software to handle specific data content defined in the chapter "[Optional supplementary fields for documents](#)".
- Preparing the e-cash register software for data exchange with the customer application as described in the chapter "[Data transfers between customer app and E-Cash Register](#)".

3.2.2 Cloud-Based E-Cash Register Application

The Cloud-Based Tax Module (FAM) provides the functionality of a Tax Unit as a service for cloud-based e-cash register (FePG) applications. The necessary steps for the development of FePG applications are as follows:

- Developing an application that meets the requirements specified in the regulation.
- Integrating the application with the service described in the chapter "[Cloud-Based Fiscal Module \(FAM\)](#)".
- Preparing the e-cash register software to handle the specific data content defined in the chapter "[Optional supplementary fields for documents](#)".



-
- Preparing the e-cash register software for data exchange with the customer application as described in the chapter "[Data transfers between customer app and E-Cash Register](#)"

3.2.3 Customer Application

The customer application delivers receipts generated in the e-receipt system to customers' mobile devices. The development of the application includes the following steps:

- Creating an application that complies with the requirements defined in the regulation.
- Implementing data structures (QR codes, NDEF messages) and cryptographic and other operations (decryption, signature verification, decompression) necessary for interpreting electronic receipts as described in the chapter "[General Description of the eReceipt Machine Interface](#)".
- Implementing calls to endpoints for querying receipts, as detailed in the chapter "[Services provided by the Receipt Repository](#)"
- Interpreting the data of electronic receipts using the descriptors detailed in the chapter "[Description of business data content \(XSD model types and elements\)](#)"
- Preparing the customer application to interpret specific data content defined in the chapter "[Optional supplementary fields for documents](#)".
- Preparing the customer application for data exchange with e-cash registers as described in the chapter "[Data transfers between customer app and E-Cash Register](#)".

3.3 Submitting a Connection Request

NAV provides a testing environment for distributors of e-cash registers and customer applications to test and verify compliance with technical requirements before the licensing procedure.

To express the intent to connect to the testing environment as an e-cash register distributor, an email must be sent to init.epg.helpdesk@nav.gov.hu with the following information:

- Distributor's tax number
- Distributor's name
- If the request is not submitted by the legal representative of the distributor, an electronically certified special authorization in the form of a private document with full probative value must be attached (a [sample](#) authorization is available on the NAV website).),
- Contact person's name
- Contact person's email address
- Contact person's phone number
- Specification of whether a hardware-based or cloud-based e-cash register is being tested
- Name of the e-cash register
- Whether the e-cash register is multi-taxpayer (yes/no)
- For hardware-based e-cash registers, a manufacturer registration CSV file and its SHA256 hash value
- For hardware-based e-cash registers, software details:
 - Software binary or its location and SHA256 hash value
 - Size of software components (in bytes)
 - Software name



- Software identifier
- Software major version number
- Date of the last software update

This information ensures the appropriate evaluation and approval process for the connection request in the testing environment.

The software data must be provided in the format and constraints specified in the XSD published on GitHub (communicationData:SoftwareType).

The manufacturer registration CSV file must contain the following data, **separated by commas**:

- IMEI number of the GSM unit embedded in the AE (imei)
- IMSI number of the SIM card embedded in the AE (imsi)
- Manufacturer name (gyarto)
- AE manufacturer model name (tipus)
- AE hardware version (hwversion)
- AE unique serial number (serialnum)
- AE software name (swnev)
- AE software version (swversion)
- Name of the GSM service provider (gsmeszolgkod)

Example of a Manufacturer Registration File

*imei,imsi,gyarto,tipus,hwversion,serialnum,swnev,swversion,gsmeszolgkod
999777111866449,999777221166446,Gyarto neve,Típus,Hardware version,Serial szám,Software neve,1,SZOLG*

3.4 Usage Requirements for Taxpayers

Before activating or transferring ownership of an e-cash register, taxpayers must possess a valid activation code.

3.5 Technologies to be implemented for connection

E-cash registers must implement the following technologies to establish a connection:

- **HTTPS** – Secure HTTP
- **REST API** – REST interface required for data services
- **XML** – Extensible Markup Language

Additionally, the following algorithms and regulations must be utilized by e-cash registers:

- **Hybrid Public Key Encryption** (RFC9180)
- **Elliptic Curve Integrated Encryption Scheme** (SECG, SEC 1)
- **AES-256 Encryption** (RFC3602)
- **RSA Encryption and Signature** (RFC8017)
- **DER Format Certificates** (RFC5280)
- **DER Format Certificate Requests** (RFC2986)
- **PEM Format Certificates** (RFC7468)
- **SHA-256** (RFC6234)



-
- **SHA3-512** (RFC6931)
 - **ECC (Elliptic Curve Cryptography)** (RFC4492)
 - **Deterministic ECDSA** (RFC6979)
 - **CMS Signed Data** (RFC5652)
 - **BASE64 Encoding** (RFC3548)
 - **GZIP Compression/Decompression** (RFC1952)
 - **XML Canonicalization** (RFC3076)

3.6 Technical requirements for E-Cash Register software

E-cash register interfaces in the live environment may only be used by e-cash registers that have been tested and approved by NAV.

E-cash register types under development, which are not yet approved, may only use the services of the test environment.

3.7 Technical requirements for customer applications

The Receipt Store interfaces in the live environment may only be used by customer applications that have been tested and approved by NAV.

Customer applications under development, which are not yet approved, may only use the services of the test environment.

3.8 Certain technical conditions for the authorization of mobile applications

The authorization process can only be initiated with an application build that has been successfully tested in the inspection environment and has been previously approved by the application store of the given platform (Apple Store or Play Store).

After authorization, only the given build may be uploaded to the application store, and in case of modifications – except for the extraordinary software modification defined in Section 19 of NGM Decree 8/2025 – a new authorization process must be initiated.



4 General description of the e-Receipt machine interface

This chapter contains technical information applicable to all services. The detailed descriptions of individual business services, including request-response structures, are provided in the chapter "[Business Services Provided by NAV for E-Cash registers](#)".

4.1 Process of eReceipt Data Services

To activate or transfer ownership of an e-cash register, an activation code must be requested. Both the e-cash register and the tax unit must have a unique identifier, the AP number.

Before commencing operation, every e-cash register must call the device registration service. The e-cash register can only be used after successful execution of this service. The registration process is detailed in the subsection "[Device registration](#)".

Each receipt, including data services for paper-based receipts and other reports, must be submitted to NAV-I immediately upon creation, in a synchronous manner, using the endpoints specified per environment in the chapter "[Accessibility of the environments](#)".

Receipts and data services must be submitted one by one through service calls; batch submission is not allowed.

Different environments require different certificate-based authentication methods, which are described in the subsection "[Authentication](#)".

Receipt data and data services must arrive signed and, where required, encrypted. The requirements for signing and encryption are detailed in the subsections "[Creating Signatures](#)" and "[Encryption](#)".

Upon submission to NAV-I, the e-cash register receives a technological response indicating success or failure. Handling failed responses is described in the chapter "[Error Handling](#)".

The business content of receipts and data services is only verified after acceptance, so no feedback on business errors is provided to the e-cash register.

4.2 Creating signatures

E-cash registers must apply a digital signature to receipts, their attachments, data service parts, cash movement receipts provided to the customer, and receipts not intended for the customer.

The digital signature ensures the authenticity, integrity, and originality of the information contained in the messages. Applying the signature allows the identification of the sender of the message and guarantees that the content of the message has not been altered during submission.

Each e-cash register must have a unique signing key for each AP number, which is requested from NAV-I during the registration process.

NAV issues the certificate to be used for signing based on an x.509 format Certificate Signing Request (CSR) compiled by the e-cash register, through the NAV-I system.

The secure storage of the signing certificate is the responsibility of the tax unit or, in the case of a cloud-based e-cash register, the FAM.



For hardware-based e-cash registers, the tax unit must store the signing certificate in a password-protected hardware key storage. The private key of the signing certificate must not leave the hardware key storage component.

The validity period of the signing certificates is two years plus 30 days from the date of issuance. Thirty days before the expiration of the signing certificate, the e-cash register must request a new signing certificate to ensure the continuous authentication of the receipts it issues. The new signing certificate can be requested via the "[Certificate Renewal](#)" xservice.

Different signing certificates must be used in the test and live environments. An e-cash register can only have one type of certificate; a test certificate cannot be replaced with a live certificate, nor can a live certificate be used for testing purposes.

Steps of the Digital Signing Process:

- The business data described in the „[Receipt Submission](#)” and [Report Reception](#)” subsections must be digitally signed.
- The digital signature must be applied to documents and reports as the final step before submission, immediately preceding the invocation of the services.

The Base64-encoded data produced as a result of the process described in the [“Receipt Submission”](#) and [“Report Reception”](#) chapters must be digitally signed before being placed into an envelope”.

- Every envelope submitted via the API must be digitally signed, regardless of whether the data to be placed in the envelope was generated with or without encryption.
- The digital signature must be executed using the signing certificate issued through NAV-I.
- The signature must be generated based on the SHA-256 hash of the produced Base64 data, using the private key associated with the signing certificate.
- The result of the signature must be stored in the "SignedDocumentEnvelopeType" / "SignedReportEnvelopeType", where:
 - envelopeData and customerEnvelopeData contain the digitally signed data in Base64 format.
 - envelopeHash contains the SHA-256 hash value of the concatenated Base64 strings stored in envelopeData and customerEnvelopeData, also in Base64 format.
 - envelopeSignature contains the Base64-encoded value of the byte sequence resulting from the digital signature.

4.3 Authentication

Access to services running on the NAV-I central system requires an authentication certificate. If a specific service can be accessed without an authentication certificate, this will be explicitly indicated for that service. NAV issues the authentication certificate through the NAV-I system based on an x.509 format Certificate Signing Request (CSR) compiled by the e-cash register.

During device registration, the e-cash register submits the CSR file, and the URL for downloading the completed certificate is provided to the e-cash register at this stage. The secure storage of the certificate is the responsibility of the tax unit or, in the case of a cloud-based e-



cash register, the FAM. The authentication certificate must be stored in a password-protected hardware key storage. The private key of the authentication certificate must not leave the hardware key storage component.

For services requiring authentication, the e-cash register must use the authentication certificate received during registration to authenticate itself to access the service.

The validity period of authentication certificates is two years plus 30 days from the date of issuance.

Thirty days before the expiration of the authentication certificate, the e-cash register must request a new authentication certificate to maintain access to the services. The new authentication certificate can be requested through the "[Certificate Renewal](#)" service.

Different authentication certificates must be used in test and live environments. An e-cash register can only possess one type of certificate; a test certificate cannot be replaced with a live certificate, nor can a live certificate be replaced with a test certificate.

An e-cash register can only be connected to either the live environment or the test environment. A live e-cash register cannot be converted into a test e-cash register, and vice versa.

4.4 Compression

Data generated by the e-cash register, such as receipts, receipt attachments, and data services, must be submitted in a compressed format. This reduces the amount of data to be sent, and the time required for submission. Compression must always be applied, regardless of the size of the data to be submitted.

The data to be compressed always includes one of the following structures:

- In the case of the "[Receipt Submission](#)":
 - CoreDocument
 - CustomerDocument
- In case of "[Report Reception](#)"
 - CoreReport
 - CustomerReport

Before compression, XML canonicalization must be performed, resulting in a normalized and consistent representation of the XML document. The purpose of canonicalization is to transform the document according to uniform standards without altering its data content. XML canonicalization is defined by the RFC 3076 ("Canonical XML Version 1.0") standard.

The data to be compressed must always include the **entire data content** of the structure, including the opening and closing tags of the data structure

That is, for example, in the case of data to be sent to the receipt store, the data to be compressed must include everything from the <CoreDocument> tag to the </CoreDocument> tag (it is important to include both the opening and closing elements):

```
<CoreDocument>
    <receiptCore>...</receiptCore>
    <receiptControl>...</receiptControl>
</CoreDocument>
```



The compression must be performed using the **GZIP format** (RFC1952), and the result of the compression is binary data.

4.5 Encryption

4.5.1 Generation of encryption key pair by the E-Cash register

The e-cash register must be capable of generating a 256-bit encryption key and an ECC encryption key pair. The ECC encryption key pair must be generated according to the ECC SECP256R1 curve (RFC5480).

4.5.2 Data encryption

Parameters for Symmetric Encryption:

- **Algorithm:** AES-256-CBC
- **Padding:** PKCS7
- **Initialization vector value:** 16 x [0]
- A new key must be generated for each document

The receipt portion of the receipt envelope must be encrypted by the e-cash register using the symmetric encryption algorithm before submission. A new AES-256 key must be generated (random 32 bytes) for each document. This key is placed in the DocumentRequest/ReportRequest and, if there is a customer envelope, also in the CustomerDocument decryptKey field in base64.

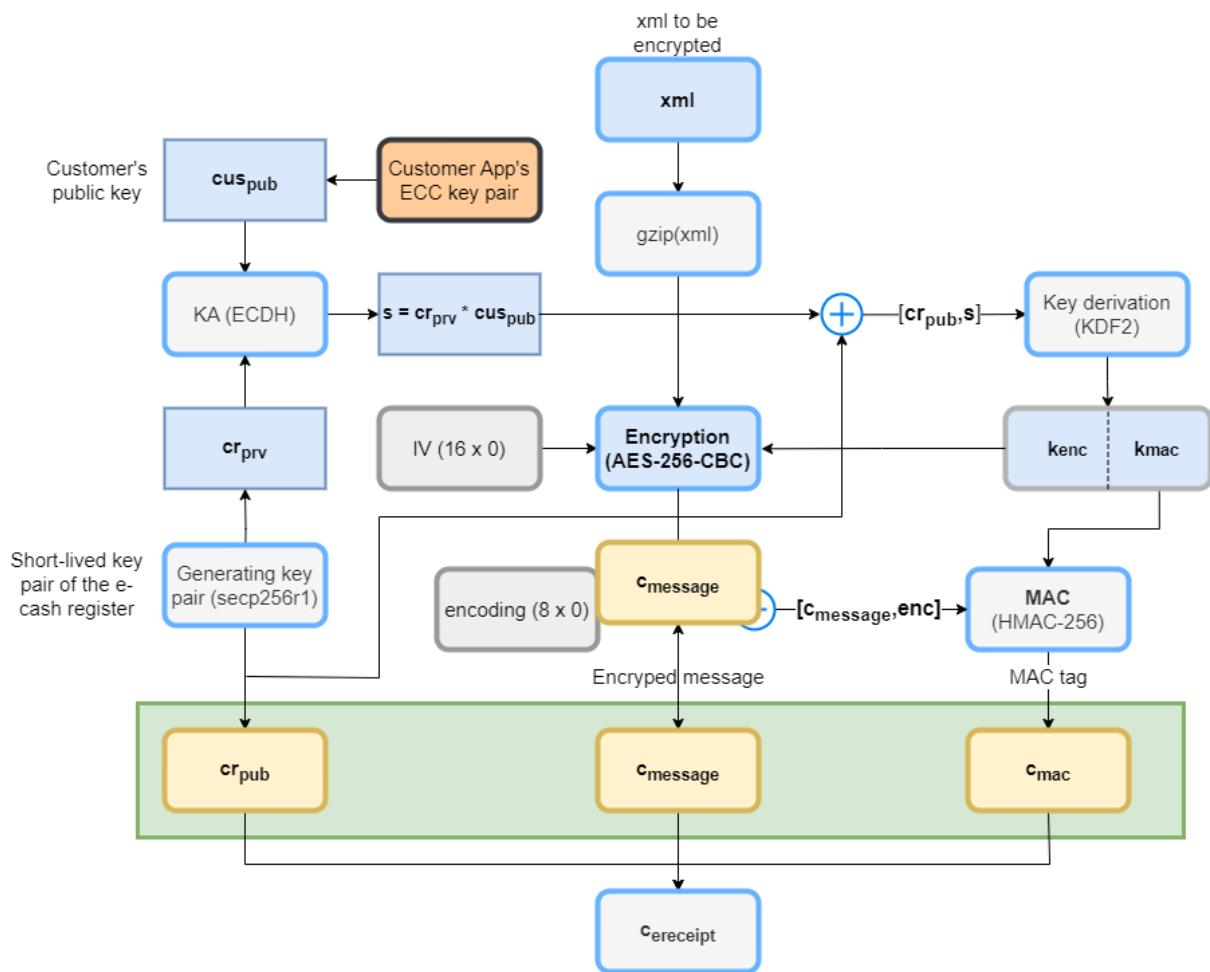
The encrypted data must always be the binary result produced by compression, as described in the "[Compression](#)" subsection. The data to be encrypted must be padded to match the AES-256 block size (PKCS#7 padding).

For encrypting customer data, the e-cash register must generate an ECC encryption key pair based on the SECP256R1 curve (RFC5480). In certain conditions, the ECC key pair may be generated by the customer application or another external party, and the public key of the customer's encryption key pair is provided to the e-cash register.

Encryption must follow the Elliptic Curve Integrated Encryption Scheme (ECIES) standard as defined in IEEE 1363a. The selected ECIES parameters are the following:

- **Key Agreement:** ECDH
- **Key Derivation Function:** KDF2 (SHA-256)
- **Symmetric Encryption Algorithm:** AES-256-CBC
- **Initialization Vector Value:** 16 x [0]
- **Padding:** PKCS7
- **MAC Algorithm:** HMAC-SHA-256
- **Compressed Public Key Format:** Yes (applies to the public key included in the result).

The application of ECIES in the e-Receipt system is illustrated in the following diagram.



Steps of the encryption process:

1. The customer presents their generated ECC key pair's public key (**cus_{pub}**) in a QR code, which is scanned by the e-cash register. For hardware-based e-cash registers, subsequent steps are executed within the **Tax Unit (AE)**, while for cloud-based e-cash registers, they are performed in the **FAM**.
2. The e-cash register generates a unique ECC key pair for each transaction. Using the customer's public key (**cus_{pub}**) and the private key (**cr_{priv}**) of the key pair generated by the e-cash register, a **Key Agreement (KA)** algorithm is used to derive a shared secret (**s**).
3. Using the **Key Derivation Function (KDF)**, symmetric keys (**k_{mac}**, **k_{enc}**) are derived from the shared secret (**s**). The input of the KDF is the **cr_{pub}** and **s** concatenated.
4. The receipt data (**xml**) is encrypted with the specified symmetric encryption algorithm (**ENC**) using the symmetric key (**k_{enc}**), resulting in the **encrypted message** (**c_{xml}**).
5. Using the encrypted message (**c_{xml}**), the encoding, and the key (**k_{mac}**), we generate the **c_{mac}** value with the specified message authentication code algorithm (MAC), by appending the encoding (8 zero bytes) to the encrypted message. The **c_{mac}** ensures that the sent message cannot be compromised.
6. The algorithm produces the **c_{receipt}** file, which contains the e-cash register's generated key pair's public key (**cr_{pub}**), the Message Authentication Code (**c_{mac}**), and the encrypted receipt data (**c_{xml}**).



Decryption is performed in the customer application. The decryption of the $c_{receipt}$ downloaded from the receipt store can only and exclusively be performed using the customer's private key (cus_{priv}).

On the customer side, the shared secret (s) must also be derived, which can be achieved using cr_{pub} and cus_{priv} . From the shared secret and the cr_{pub} , the k_{mac} and k_{enc} can be derived in the same way. With the symmetric key k_{enc} , the c_{xml} can be decrypted. Using k_{mac} , c_{xml} , and encoding, the MAC can be generated, which must match the c_{mac} for the $c_{receipt}$ to be considered authentic.

Using the symmetric key k_{enc} , the encrypted message c_{xml} can be decrypted. With k_{mac} and c_{xml} , the MAC can be generated, which must match c_{mac} for the $c_{receipt}$ to be considered authentic.

4.6 Generating QR codes and NDEF (NFC) data packages by the E-Cash register

4.6.1 Generating input QR code by the E-Cash register

According to Annex 2, Part A, point 8 of the Regulation, the e-cash register must be capable of scanning a QR code and interpreting its data content regarding the following information:

- a) Receiving an encryption key from the customer's device
- b) Receiving the customer's invoice request
- c) Receiving the customer's data in case of an invoice request
- d) Receiving the payment method intended to be used by the customer
- e) Receiving the email address specified by the customer
- f) Receiving a coupon code (discount code) the customer wishes to redeem
- g) Receiving data regarding a percentage-based or fixed amount tip specified by the customer
- h) Receiving and interpreting the loyalty program ID the customer wishes to use
- i) Receiving the bank account number or secondary identifier intended to be used by the customer in the Instant Payment System (qvik)
- j) Receiving the timestamp of the QR code creation

This functionality of e-cash registers enables the transmission of customer preferences related to purchases and their documentation in a standardized format to the e-cash register.

The QR code intended for scanning by the e-cash register is typically generated by the customer application. Every e-cash register must be capable of scanning one or more input QR codes during a sales transaction and correctly interpreting their contents, even if the operator of the e-cash register does not intend to use these features.

The purpose of this functionality is to allow customers to conveniently and uniformly communicate their preferences regarding payments and receipts in a format that all e-cash registers can interpret.

The QR code content can also be transmitted via NFC as a textual NDEF record from the customer application to the e-cash register, provided the e-cash register is equipped with the appropriate peripherals.

The data must be created in the following compact format and encoded into the QR code:

- UTF-8 encoded character string without line breaks.
- The first character is the digit „1” indicating that this is a QR code generated in a customer application.
- Data fields (ASCII 24) must be separated by the „|” character.



- Any „|” character within the data must be escaped using the „\” sequence (backslash). The escape character itself must also be escaped with „\\” (e.g., „\\|”).
- If the data contains a line break (e.g., a two-line address), the line break must be replaced with the „\\n”character sequence.
- The first character of each data field identifies the respective field.
- UTF-8 encoding must also be specified in the QR code header.

The order of data included in the QR code is fixed according to the sequence outlined below. Each piece of data must be represented as a string, even if it could be interpreted as a number. Including any specific piece of data is optional, but all fields may be included if necessary. Certain data fields can appear multiple times, as indicated in the table. For customer data, the corresponding element names from the invoice XSD schema are noted in parentheses.

Mandatory Annotations to be interpreted by the E-Cash Register

Serial nr	Identifier	Description of data to be mandatorily interpreted
1.	K	The public part of the encryption ECC key pair in compressed format (33 bytes), containing only the key bytes, encoded in Base64
2.	I	Indicates the type of receipt (invoice or receipt) the customer wishes to receive for the given transaction and its desired format. Value set: P – The customer wishes to receive a paper invoice I – The customer wishes to receive an electronic invoice R – A paper copy of the receipt E – E-receipt (default option)
3.	N	In the case of an invoice request by the customer, the name of the customer to be included on the invoice.
4.	S	The VAT status of the customer (customerVatStatus). Value set: D – Domestic P – Non-VAT-registered individual O – Other
5a.	X	The customer's domestic tax number to be included on the invoice, in the format 12345678-1-12 or 12345678 (customerTaxNumber).
5b.	U	The customer's EU VAT number to be included on the invoice. (communityVatNumber)
5c.	H	The customer's non-EU (third country) tax number to be included on the invoice. (thirdStateTaxId)
6.	G	If the customer is a member of a VAT group, the group member's tax number in the format 12345678-1-12 or 12345678 (groupMemberTaxNumber).
7.		The customer's address to be included on the invoice. (simpleAddress)
7/1	C	Country code according to the ISO 3166 alpha-2 standard. (countryCode)
7/2	R	Region code (if applicable in the given country) according to the ISO 3166-2 alpha-2 standard (region).
7/3	Z	Postal Code. (postalCode)
7/4	Y	City. (city)



Serial nr	Identifier	Description of data to be mandatorily interpreted
7/5	A	Additional Address Details. (additionalAddressDetail)
8.	M	The email address through which the seller can contact the customer to provide further information related to the transaction. The authenticated transfer of receipt data is conducted through the receipt store.
9.	P	The payment method the customer intends to use. Value set: C – cash B – bank card S – SZÉP card A – Instant Payment System (qvik)
10.	F	The bank account number or secondary account identifier the customer intends to use during the Instant Payment System transaction (qvik).
11.	V	The coupon code (discount code) the customer wishes to redeem. It can be included multiple times consecutively.
12.	T	The tip offered by the customer. Possible values: <ul style="list-style-type: none">• An integer representing the tip amount in forints.• N% notation, where N indicates the tip as a percentage of the total amount to be paid.
13.	L	The loyalty program ID the customer intends to use.
14.	Q	If the data the customer wishes to share is distributed across multiple QR codes, this field indicates the position of the current QR code and the total number of QR codes to be scanned. Notation: serial number/total. For example, in a series of three QR codes, the second QR code would be marked as „2/3”. This marker must be included as the first identifier in each partial QR code.
15.	D	The timestamp of the QR code generation in Coordinated Universal Time (UTC), represented in unix time format, encoded in Base64, and spanning 8-16 characters. Example: <ul style="list-style-type: none">• Date: May 7, 2024, 13:32:29• Unix time (decimal): 1715088749• Hexadecimal: [0x66, 0x3A, 0x2D, 0x6D]• Base64: ZjotbQ==

Constraints and notes:

The following constraints must primarily be considered by customer application developers to ensure that the input QR code for the e-cash register does not contain logically conflicting customer requests or irrelevant data for the given situation.

1. Customer data (name, tax number, address) may only be included if an invoice request is also indicated (**I** marker).
2. Only one of the data fields 5a, 5b, or 5c from the table may be included.



3. If the customer is a member of a domestic VAT group, the group tax number must be used on the invoice as the customer's tax number according to the law. In such cases, there is typically a need to display the group member's tax number on the invoice as well (**G** marker).
4. The bank account number or secondary account identifier intended for use with the Instant Payment System (qvik) may only be included if the **P** marker has the value **A** or if the **P** marker is not included.
5. If the QR code includes an encryption key (the public part of the generated key pair), the timestamp of the QR code generation must also be included.
6. A QR code must not exceed **Version 12**, meaning it can consist of a maximum of **65x65 modules** and must have at least an **M** error correction level. With UTF-8 encoding, this allows for a maximum of **287 bytes** of characters. If the data to be transmitted exceeds 287 bytes, the QR code must be split using the **Q** marker. Data splitting can only occur at field boundaries.
7. The e-cash register must be prepared to handle cases where the QR codes in a multi-code data transmission are scanned out of sequence.

The e-cash register examines the QR code's generation timestamp. If it is older than **5 minutes**, it must assume that the presented QR code was not uniquely generated by a customer application but was instead presented as a static image or in printed form. The AE must indicate the staleness of a search key with a timestamp older than 5 minutes in the corresponding field of the submitted XML.

It is not prohibited to present the QR code as an image or in printed form to the e-cash register. For instance, a practical use case could involve a procurement officer providing the customer company's data in a QR code while always requesting a paper invoice and not providing an encryption key.

Non-mandatory annotations (Recommendations)

The interpretation of the following annotations is not required by law. However, it is important to ensure that the same data is marked consistently by all customer applications. To this end, this Developer Documentation summarizes, in the table below (which will be continuously expanded), the optional data that can be included in the customer QR code and optionally interpreted.

Suggestions for adding data not yet included should be submitted via the following website:
<https://github.com/nav-gov-hu/eRECEIPT/discussions>.

Optional annotations must be included in the QR code **after the mandatory annotations**.

Serial nr	Identifier	Description of optionally interpretable data.
1.	E01	Indicates whether the customer requests the order for on-site consumption or takeaway. Value set: L – On-site consumption T – Takeaway



Examples to assist interpretation:

The examples provided here contain fictitious data.

Example 1

The customer only wishes to share an encryption key, with no additional data. The ECC key to be shared is as follows:

```
-----BEGIN PUBLIC KEY-----  
MDYwEAYHKoZIzj0CAQYFK4EEAAoDgADPAuX18eJbDt3BNJ5XegLwt1kezR4XrZcNkmHDWwpVus=  
-----END PUBLIC KEY-----
```

The QR code was generated on May 31, 2023, at 9:01:05 AM Hungarian time (GMT+2, due to daylight saving time).

In this case, the character string serving as the basis for the QR code is as follows:

```
1|KAzwLl9fHiWw7dwTSeV3oC8LZZHs0eF62XDZJhw1sKVbr|DZHbwsQ==
```

QR code generated:



Example 2

The customer does not provide a key. The customer requests an electronic invoice to be sent to example@donotuseit.org. The customer's name is Első Magyar Általános Művek Zrt., which is a domestic VAT taxpayer and a member of a VAT group. The VAT group's tax number is 12345678-5-99, and the customer also requests the group member's tax number (11223344-4-90) to be included on the invoice. The customer's address is 9876 Hosszúnévfalva, Kossuth Lajos utca 69. The customer also indicates that they wish to pay in cash.

In this case, the character string serving as the basis for the QR code is as follows (with line breaks and spaces at the beginning of some lines for better readability; these are not included in the QR code):

```
1|II|NELső Magyar Általános Művek Zrt.|SD|X12345678-5-99|G|11223344-4-90|CHU|Z9876|YHosszúnévfalva|AKossuth Lajos utca 69.|Mexample@donotuseit.org|PC
```



QR code generated:



Example 3

The customer provides the following encryption key:

```
-----BEGIN PUBLIC KEY-----  
MDYwEAYHKoZIzj0CAQYFK4EEAAoDgADPAuX18eJbDt3BNJ5XegLwt1kezR4XrZcNkmH  
DWwpVus=  
-----END PUBLIC KEY-----
```

The customer does not request an invoice (this does not need to be marked separately) and wishes to pay using the Instant Payment System (qvik), utilizing the secondary identifier **john.doe@thisimysecondary.com**. During the payment process, they offer a 12% tip.

The QR code was generated on March 1, 2023, at 12:45:15 AM Hungarian time (due to daylight saving time, GMT+1).

In this case, the character string serving as the basis for the QR code is as follows:

```
1|KAzwLl9fHiWw7dwTSeV3oC8LZZHs0eF62XDZJhw1sKVbr|PA|F|john.doe@thisimy  
secondary.com|T12%|DY/6gGw==
```

QR code generated:





Example 4

The customer provides the following encryption key:

```
-----BEGIN PUBLIC KEY-----  
MDYwEAYHKoZIzj0CAQYFK4EEAAoDIgADPAuX18eJbDt3BNJ5XegLwt1kezR4Xr  
ZcNkmHDWwpVus=  
-----END PUBLIC KEY-----
```

The customer requests a paper invoice, which they wish to have issued in their own name (a natural person not subject to VAT) with the following details:

Jürgen Fussball, 303 E Wacker Drive Suite 1200 Chicago, IL 60601.

The customer indicates their intention to pay by bank card. During their purchase, they wish to use the following coupons: 00000001, 00000002, 00000003, 00000004, 00000005, 00000006, 00000007, 00000008, 00000009, 00000010, 00000011, 00000012, 00000013, 000000+1.

The customer also wishes to use a loyalty card for their purchase, with the identifier ABCDEFGHIJKLMNOPQRSTUVWXYZ. The customer requests the food for takeaway, which, fortunately, the customer application they use can indicate in the QR code.

The QR code is generated on March 26, 2023, at 2:00 AM Hungarian time, during daylight saving time (precisely when clocks are set forward to 3:00 AM)

In this case, the character string serving as the basis for the QR code is as follows:

```
1|KAzwLl9fHiWw7dwTSeV3oC8LZZHs0eF62XDZJhw1sKVbr|IP|NJürgen  
Fussball|SP|CUS|RUS-IL|Z60601|YChicago|A303 E Wacker Drive Suite  
1200|PB|V00000001|V00000002|V00000003|V00000004|V00000005|V00000006|  
V00000007|V00000008|V00000009|V00000010|V00000011|V00000012|V0000001  
3|V000000+1|LABCEFGHIJKLMNOPQRSTUVWXYZ|D ZB+ZEA==|E01T
```

Generated QR Code:



Given the amount of data to be represented, this QR code would not be readable on a significant portion of devices. Therefore, the data is displayed across two QR codes. This essentially means generating two independently meaningful QR codes containing disjoint data, with both including the "C" marker.



The character string serving as the basis for the 1st QR code is as follows:

1|Q1/2|KAzwLl9fHiWw7dwTSeV3oC8LZZHs0eF62XDZJhw1sKVbr|IP|NJürgen
Fussball|SP|CUS|RUS-IL|Z60601|YChicago|A303 E Wacker Drive Suite
1200|PB

The first QR code:



The character string serving as the basis for the 2nd QR code is as follows:

1|Q2/2|V00000001|V00000002|V00000003|V00000004|V00000005|V00000006|V
00000007|V00000008|V00000009|V00000010|V00000011|V00000012|V00000013
|V000000+1|LABCDEFGHIJKLMNOPQRSTUVWXYZ|D ZB+ZEA==|E01T

The second QR code:





4.6.2 Formation of the output QR code for the E-Cash register

According to the legislation, the e-cash register must print a duplicate receipt if the customer requests it.

The QR code at the end of the duplicate receipt must, according to the legislation, contain the following data:

1. The AP number,
2. The serial number of the issued receipt,
3. The core tax number of the operator,
4. The date and time of issuance,
5. The gross total amount of the receipt,
6. The data required for downloading and interpreting the issued receipt, as specified in the Developer Documentation.

The data specified in point 6 are as follows:

- The private key of the customer encryption key pair generated by the e-cash register for encrypting the receipt data sent to the receipt repository, provided the key pair was generated by the e-cash register.
- The search key generated by the e-cash register or the customer application.
- The time of QR code generation by the e-cash register, or the date indicated in the data package containing the customer encryption key received from the customer application (search date).
- The signature of the QR code.
- The serial number of the certificate used for signing the QR code.
- The AP number and the operator's tax identification number form part of the serial number of the document, therefore they do not need to be included in a separate field.

The data content of the QR code can also be read by the customer application via NFC in the form of a textual NDEF record, provided the e-cash register is equipped with the appropriate peripheral.

The data must be created in the following compact format and encoded into the QR code:

- UTF-8 encoded character string without line breaks.
- The first character is the digit "2," indicating that this is an e-cash register output QR code.
- Data fields must be separated by the "|" character.
- Any "|" character within the data must be escaped with the "\\" (backslash) character (e.g., "\\|"). The escape character itself must also be escaped (e.g., "\\\").
- If the data contains a line break, it must be replaced by the "\\n" character sequence.
- The first character of each data field is the identifier of that field.
- UTF-8 encoding must also be specified in the QR code header.

All data must be represented as character strings (strings), even if they could also be interpreted as numbers.



Serial nr	Identifier	Data description
1.	B	The serial number of the issued receipt.
2.	D	The timestamp of the search key generation in Coordinated Universal Time (UTC), in Unix time format, encoded in base64, with a length of 8-16 characters. Example: <ul style="list-style-type: none">• Date: May 7, 2024, 13:32:29• Unix time (decimal): 1715088749• Hexadecimal: [0x66, 0x3A, 0x2D, 0x6D]• Base64: ZjotbQ==
3.	I	To represent the issuance date and time of the receipt in Coordinated Universal Time (UTC) in Unix time format with base64 encoding (8-16 characters).
4.	T	The gross total amount of the receipt in Hungarian Forints (HUF) should be represented without thousand separators and with two decimal places after a decimal point.
5.	S	The search key generated by the e-cash register or the customer application must be represented using base64 encoding.
6.	K	The private key of the encryption key pair (used for decoding), if the key pair was generated by the e-cash register, must be represented exclusively as the key's bytes, encoded in base64. If the encryption key pair was generated by the customer application, this field must be omitted.
7.	G	The signature of the QR code.
8.	C	The unique serial number of the certificate used for signing the QR code, ensuring its validity and authenticity.

In the customer application, with the help of the field starting with the **K** marker, it can be identified whether the e-cash register has taken over a key from the customer application for the given receipt, or if it was generated by the e-cash register. If a field with the **K** marker exists, then the encryption key was generated by the e-cash register. If there is no **K** marker, the customer application must search for the encryption key in its own storage based on the search key.

The steps of the digital signature process are as follows:

- The QR code data in positions 1-4 (B, I, D, T, and S markers), including the separators and field identifiers, must be digitally signed (excluding the first part of the string in the QR code, except for 2| and the separator before the K marker).
- The encryption key is not part of the data to be signed.
- The digital signature must be performed using the signing certificate issued by the tax authority (NAV) for the e-cash register.
- The signature must be generated based on the string to be signed, using the private key.
- The digital signature must be created according to ECDSA
- The digital signature must be compacted in the following manner:
 - The "r" and "s" values from the signature result must be extracted.
 - Both values must be obtained in big-endian byte order, and if necessary, padded to 32 bytes with zeros.



- The resulting byte arrays must be concatenated (first r, then s), which will result in a 64-byte long compact signature.
- The base64 encoded value of the compact digital signature must be placed in the field starting with the G marker.

To verify the signature, the signing certificate of the e-cash register issuing the receipt will be provided by the receipt query endpoint of the receipt repository to the customer application.

Example for interpretation:

The example provided here contains fictitious data. Since the standard ECDSA provides a non-deterministic result, the signature generated with the sample data to be signed and the private key cannot be reproduced. Nevertheless, the verification of the signature must be successful. The electronic receipt, whose data is contained in the QR code, can be described with the following details:

E-cash register AP number: B99912345

Serial number of the issued receipt: NY-B999123456/98765432/0099/00004

Operator's tax number (core number, 8 digits): 98765432

Date and time of receipt issuance: October 2, 2023, 13:45:56 Hungarian time (DST, GMT+2)

Gross total of the receipt: 13,470 HUF

The **signing private key** of the e-cash register:

```
-----BEGIN EC PRIVATE KEY-----
MDECAQEEIMmvqdhFunUWalwhV2ex1pNOUMPbNuibEnuKYissD2choAoGCCqGSM49
AwEH
-----END EC PRIVATE KEY-----
```

The **public key** of the signing key pair:

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEYP7UuiVanTHJYet0xjVtaMBJuJI7
Yfps5mliLmDyn7Z5A/4QCLi8maQa6elWKLxk8vGyDC1+n1F3o8KU1EYimQ==
-----END PUBLIC KEY-----
```

The **private key** of the encryption key pair (used for decoding):

```
-----BEGIN EC PRIVATE KEY-----
MDECAQEEIIjZUbDcu40upJ+P1/TJZbDj9yK1TXC1PuT/pvpAI6S+oAoGCCqGSM49
AwEH
-----END EC PRIVATE KEY-----
```

The **public key** of the encryption key pair:

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEKR7bXJnV2Asaz6ng2VXHrdSBy0JM
Sff1tng9t1E2wcqRr0naYvPsh3OwL+VrlvUT9c5Hcs5Arc6OFm/SjmGk6g==
-----END PUBLIC KEY-----
```

Raw Encryption Private Key Extracted from the PEM Format, Encoded in Base64:

AIjZUbDcu40upJ+P1/TJZbDj9yK1TXC1PuT/pvpAI6S+

Raw Public Key Extracted from the PEM Format, Encoded in Base64:

Aike21yZ1dgLGs+p4N1Vx63UgctCTEnxdbZ4PbdRNshK



Search key generated from the raw public key: binary content encoded in Base64:
nL6P7YkWIS98by3uo8wvQfrqzmBbICOgMtW8PPiMuPU=

The serial number of the certificate used for the signature in hexdump format:
10:c4:f2:b0:7c:2c:b0:84:aa:36:70:4c:42:2e:e5:95:a4

The serial number of the certificate used for the signature, encoded in Base64:
EMTyshwsSISqNnBMQi7llaQ=

The QR code was generated on October 2, 2023, at 13:45:53 AM Hungarian time (due to summertime, GMT+2).

Date in Base64:
ZRqtcQ==

Data to be signed:

BNY-B999123456/98765432/0099/00004|DZRqtcQ==|IZRqtdA==
|T13470.00|SnL6P7YkWIS98by3uo8wvQfrqzmBbICOgMtW8PPiMuPU=

The hash of this sha-256 is written in hex:
3ad0ca19d0eaecc6c18a748decd8d976b28d1b34de9793f778821b8595885e7d

The values of the generated signature ‘r’ and ‘s’ in hex:

r:a7a381af732fa6742beab0ea54c260b2d9c52fd4922b49436ab64b50f43a4f90
s:3073b487d521c75c4c9216660b48c558012c60f6d98a53fc63f001b0c87bc0b9

The compact signature formed from this:

p6OBr3MvpnQr6rDqVMJgstnFL9SSK01DarZLUPQ6T5Awc7SH1SHHXEySFmYLSMVYA
Sxg9tmKU/xj8AGwyHvAuQ==

The character string serving as the basis for the QR code is as follows:

2|BNY-
B999123456/98765432/0099/00004|DZRqtcQ==|IZRqtdA==|T13470.00|SnL6P7YkWIS98by3u
o8wvQfrqzmBbICOgMtW8PPiMuPU=|KAIjZUbDcu40upJ+P1/TJZbDj9yK1TXC1PuT/pvpAI6S+|Gp6
OBr3MvpnQr6rDqVMJgstnFL9SSK01DarZLUPQ6T5Awc7SH1SHHXEySFmYLSMVYASxg9tmKU/xj8AGw
yHvAuQ==|CEMTysHwsSISqNnBMQi7llaQ=

Generated QR Code:



4.6.3 Generation of the E-Cash register signature verification QR code

According to the regulations, the e-cash register must have a function that allows it to digitally sign a specific input data set and display it on the screen or print it.



The signature verification QR code must contain the following data:

1. The AP number,
2. The core tax number, and in the case of a dual-operator e-cash register, the core tax numbers of both operators,
3. The IMSI number of the AE SIM,
4. The IMEI number of the AE modem.

For cloud-based e-cash registers, FAM includes an internal technical IMSI and IMEI number in the data set.

The data must be formatted in the following compact format and encoded into the QR code:

- A UTF-8 encoded character sequence without line breaks.
- The first character must be the digit „3” indicating that this is an e-cash register signature verification QR code.
- Data fields must be separated by the „|” character.
- If the „|” character appears within a data field, it must be escaped with a backslash („\|”). The escape character itself must also be escaped (e.g., „\\|”).
- If the data contains line breaks, they must be replaced with the „\\n” sequence.
- The first character of each data field must be its identifier.
- UTF-8 encoding must be specified in the QR code header.

All data must be represented as character strings, even if they could be interpreted as numeric values.

Serial nr	Identifier	Data description
1.	D	The date and time of QR code generation in Coordinated Universal Time (UTC), represented in Unix time format, encoded in base64 with a length of 8-16 characters. Example: May 7, 2024, 13:32:29 → 1715088749 (decimal) → [0x66, 0x3A, 0x2D, 0x6D] (hexadecimal) → ZjotbQ== (base64)
2.	A	AP number of the e-cash register
3.	N	The core tax number of the e-cash register operator (the first 8 digits of the tax number). For a dual-operator e-cash register, the core tax numbers of both operators must be listed by repeating the field.
4.	E	The IMEI number of the Fiscal Unit's modem
5.	S	The IMSI number of the Fiscal Unit's SIM card
6.	G	The digital signature of the QR code
7.	C	The serial number of the certificate used to sign the QR code



The steps of the digital signature process:

- The digital signature must be applied to the data located at positions 1-5 in the QR code (indicated by D, A, N, E, and S), including separators and field identifiers (excluding the initial „3|” in the QR code string and without including the separator before D).
- The digital signature must be executed using the signing certificate issued by NAV to the e-cash register.
- The signature must be generated from the string to be signed, using the private key.
- The digital signature must be created according to ECDSA.
- The digital signature must be converted into a compact format, which is structured as follows:
 - Extract the **r** and **s** values from the signature result.
 - Both values must be extracted in big-endian byte order, and, if necessary, padded to 32 bytes with zeros.
 - The extracted byte arrays must be concatenated (r followed by s), resulting in a 64-byte signature.
 - This compact signature must then be base64-encoded.
 - The base64-encoded compact digital signature must be placed in the G-labeled field of the QR code.

Example for Understanding:

The following example contains fictional data. Since the standard ECDSA provides a non-deterministic result, the signature generated with the sample data to be signed and the private key cannot be reproduced. Nevertheless, the verification of the signature must be successful.

The electronic receipt, whose data is included in the QR code, can be described as follows:

AP number of the e-cash register: B99912345

The core tax number of the e-cash register operator (8 digits): 98765432

Date and time of QR code generation: October 2, 2023, at 13:45:56 Hungarian time (Daylight Saving Time, GMT+2)

IMEI number of the Fiscal Unit's modem: 111110123456789

IMSI number of the Fiscal Unit's SIM card: 111119876543210

The private key of the cash register's signing key pair:

```
-----BEGIN EC PRIVATE KEY-----  
MDECAQEEIMmvqdhFunUWa1whV2ex1pNOUMPbNuibEnuKYisSD2choAoGCCqGSM49  
AwEH  
-----END EC PRIVATE KEY-----
```

The public key of the signing key pair:

```
-----BEGIN PUBLIC KEY-----  
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEYP7UuiVanTHJYet0xjVtaMBJuJI7  
Yfps5mlilMmDyn7Z5A/4QCLi8maQa6e1WKLxk8vGyDC1+n1F3o8KU1EYimQ==  
-----END PUBLIC KEY-----
```

The encryption key pair (used for decryption) private key:

```
-----BEGIN EC PRIVATE KEY-----  
MDECAQEEIIjZUbDcu40upJ+P1/TJZbDj9yK1TXC1PuT/pvpAI6S+oAoGCCqGSM49  
AwEH  
-----END EC PRIVATE KEY-----
```



The encryption key pair public key

```
-----BEGIN PUBLIC KEY-----  
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEKR7bXJnV2Asaz6ng2VXHrdSBy0JM  
SfF1tng9t1E2wcqRr0naYvPsh30wL+VrlvUT9c5Hcs5Arc60Fm/SjmGk6g==  
-----END PUBLIC KEY-----
```

Serial number of the certificate used for the signature in hexdump format:

10:c4:f2:b0:7c:2c:b0:84:aa:36:70:4c:42:2e:e5:95:a4

Serial number of the certificate used for signing with base64 encoding:

EMTysHwssISqNnBMQi711aQ=

Date and time of QR code generation:

October 2, 2023, at 13:45:56 Hungarian time (Daylight Saving Time, GMT+2)

Date in base64:

ZRqtA==

The data to be signed will be

DZRqtA==|AB99912345|N98765432|E111110123456789|S111119876543210

The generated signature r and s values in hexadecimal notation:

r: 519d4bc9ac046f1060ef85d69d5c8cadf7b92c05a171ce9b68dbde29e718e20
s: 15497d1e3f9be1653ef24b57746e122593256a44e66e9b7935e456414b3751e2

The compact signature derived from this:

UZ1LyprARvEGDvhadXIyt97ksBaFxzpto294p5xjiAVSX0eP5vhZT7yS1d0bhIlkyVqROZum3k15FZBSzdR4g==

The character sequence serving as the basis for the QR code will be:

3|DZRqtA==|AB99912345|N98765432|E111110123456789|S111119876543210|GUZ1LyprARvEGDvhadXIyt97ksBaFxzpto294p5xjiAVSX0eP5vhZT7yS1d0bhIlkyVqROZum3k15FZBSzdR4g==|CEMTysHwssISqNnBMQi711aQ=

The generated QR code:



4.6.4 Interpretation of Cloud-based e-Cash Register Activation QR Code

A cloud-based e-cash register (FePG) instance can be requested through the ePG Portal. After a successful creation, the ePG Portal displays a QR code to the operator, which contains a short-lived token necessary for activating the cloud-based e-cash register application.

The FePG short-lived token QR code contains the following data:

1. The AP number
2. The login password of the FAM instance
3. The short-lived code required for requesting the client authentication certificate of the FePG application

The data appears in the following compact format in the QR code displayed on the ePG Portal:

- UTF-8 encoded character string without line breaks.
- The first character is the digit „4” indicating that this is a QR code containing a short-lived token for a cloud-based e-cash register.
- Data fields are separated by the „|” character.
- Any „|” character within the data is escaped with the „\|” (backslash) character („\\|”). The escape character itself is escaped with another backslash („\\\\|”).
- If there is a line break in the data, it is replaced with the „\\n” character sequence.
- The first character of each data field is its identifier.
- UTF-8 encoding is also specified in the QR code header.

All data is represented as a string, even if it could be interpreted as a number.

Serial No.	Identifier	Data Description
1.	D	The date and time of QR code generation in Coordinated Universal Time (UTC), in Unix time format, base64 encoded (8-16 characters). Example: May 7, 2024, 13:32:29 → 1715088749 (decimal) → [0x66, 0x3A, 0x2D, 0x6D] (hexadecimal) → ZjotbQ== (base64)
2.	U	The AP number of the cloud-based e-cash register instance
3.	P	The login password of the cloud-based e-cash register application
4.	T	The short-lived code required for requesting the client authentication certificate of the FePG application

In the inspection and the live environment, the FAM services must be called at different server addresses, and the environment is identified based on the first letter of the AP number read from the activation QR code:

- T: inspection (test)
- C: live (production)

The URLs belonging to each environment can be found in the chapter [Accessibility of the environments](#).



Example for Interpretation:

The example below contains fictional data.

AP number of the eCash register: C99912345

Login password: Z5FMGKvZ9FwX

The short-lived code for requesting the client authentication cert:
YAApDs3ubNd8GNKBxpNYkr6kbVxXsDHR

The date and time of QR code generation: October 2, 2023, 13:45:56 Hungarian time, during daylight saving time (GMT+2)

Date in base64:

ZRqtdA==

The character sequence serving as the basis for the QR code will be:

4|DZRqtdA==|UC99912345|PZ5FMGKvZ9FwX|TYAApDs3ubNd8GNKBxpNYkr6kbVxXsDHR

The generated QR code:



4.6.5 Creating an NDEF record for input code confirmation

E-cash registers can optionally – if equipped with a suitable peripheral – receive the "input QR code" content from the customer application via NFC in a text-based NDEF record. Since NFC allows bidirectional data flow, e-cash registers must confirm the received code back to the customer application.

The confirmation NDEF record must contain the following data:

1. The result of the data reception, which can be either successful or erroneous.
2. The SHA-256 checksum of the received full message.

The data must be created in the following compact format and encoded into the QR code:

- UTF-8 encoded character sequence without line breaks.
- The first character is the number „5” indicating that this is an NDEF record confirming the receipt of customer data.
- Data fields must be separated by the „|” character.



- Any „|” character appearing within the data must be escaped using the „\|” (backslash) character „\|”). The escape character itself must also be escaped (e.g., „\\|”).
- If there is a line break in the data, it must be replaced with the „\n”character sequence.
- The first character of each data field is the identifier of that field.

All data must be represented as a string, even if it could be interpreted as a number.

Serial Number	Identifier	Data Description
1.	R	<p>Response code indicating successful receipt or an error as follows:</p> <ul style="list-style-type: none">• OK: Successful reception• E0000: Syntactical error, the provided code is not interpretable• E0500: Other error• E0501: Customer data transferred without indicating an invoice request• E0502: Only one tax number can be included (only one of data fields 5a, 5b, and 5c can be present at the same time)• E0503: AFR bank account number or secondary identifier cannot be included if the payment method (P identifier) is not "A".• E0504: Key transfer without a timestamp• E0505: The encryption key is not in the correct format• E0506: Invalid timestamp
3.	H	SHA-256 checksum of the received message, encoded in base64.

Example for Interpretation:

The example below contains fictional data.

The content of the input code provided by the customer application:

1|KAzwL19fHiWw7dwTSeV3oC8LZZHs0eF62XDZJhw1sKVbr|PA|Fjohn.doe@thisimysecondary.com|T12%|DY/6gGw==

Successful reception of the code.

The NDEF message character string will be as follows:

5|ROK|H0dnxPrCuEo1aEFc28c41jqPx8+LjYP17hhF0kn+L0v4=

4.6.6 NDEF record creation for requesting a receipt envelope

E-cash registers equipped with NFC peripherals may optionally allow the full receipt envelope to be downloaded via a textual NDEF message. The customer application can only request the receipt whose serial number matches the NDEF record read from the output QR code content.



The receipt request NDEF record must contain the following data:

1. The serial number of the requested receipt.

The data must be created in the following compact format and encoded into the QR code:

- UTF-8 encoded character sequence without line breaks.
- The first character is the digit „6”, indicating that this is a receipt envelope request NDEF record.
- Data fields must be separated by the „|” character.
- Any „|” character within the data must be escaped with a „\” (backslash) character („\\|”). The escape character itself must be escaped („\\\\|”).
- If the data contains a line break, it must be replaced with the „\\n” character sequence.
- The first character of each data field is its identifier.

All data must be represented as a string, even if it could be interpreted as a number.

Serial nr	Identifier	Data description
1.	B	Serial number of the requested receipt

Example for Interpretation:

The example below contains fictional data.

The customer application requests the following e-receipt envelope:

NY-B999123456/98765432/0099/00004

The NDEF message character string will be as follows:

6|BNY-B999123456/98765432/0099/00004

4.6.7 NDEF record creation for receipt envelope

If the customer application requests the full message envelope content via NFC communication, the e-cash register provides it in a designated NDEF record. This transmission does not exclude the possibility of downloading the envelope from the Receipt Repository; direct reading is an optional feature.

The receipt envelope—especially if it includes an attachment—may not fit within the maximum NDEF message size supported by the NFC peripheral connected to the e-cash register. The e-cash register must indicate this limitation in the response message to the customer application.

The receipt envelope NDEF record must contain the following data:

1. The result of data reception, which can be either successful or erroneous.
2. The receipt envelope (documentEnvelope or reportEnvelope).
3. The serial number of the signing certificate used by the e-cash register.

The data must be created in the following compact format and encoded into the QR code:

- UTF-8 encoded character sequence without line breaks.
- The first character is the digit „7”, indicating that this is a receipt envelope NDEF record.



- Data fields must be separated by the "|" character.
- Any „|” character within the data must be escaped with a „\” (backslash) character („\\|”). The escape character itself must be escaped („\\\\|”).
- If the data contains a line break, it must be replaced with the „\\n” character sequence.
- The first character of each data field is its identifier.

All data must be represented as a string, even if it could be interpreted as a number.

Serial nr	Identifier	Data description
1.	R	Response code indicating successful processing or an error as follows: <ul style="list-style-type: none">• OK: successful processing• E0000: Syntactic error, the provided code is not interpretable• E0700: Other error• E0701: The receipt size exceeds the maximum NDEF record size
2.	V	The XSD schema version used when submitting the receipt ("1.0").
3.	D	Receipt envelope, identical to the receipt data service XML submitted to the Receipt Repository: <ul style="list-style-type: none">• For sales receipts: documentRequest / documentEnvelope / envelopeData• For reports: reportRequest / reportEnvelope / envelopeData
4.	U	Customer envelope, identical to the receipt data service XML submitted to the Receipt Repository: <ul style="list-style-type: none">• For sales receipts: documentRequest / documentEnvelope / envelopeData• For reports: reportRequest / reportEnvelope / customerEnvelopeData
5.	H	SHA256 hash fingerprint of the envelopes, identical to the receipt data service XML submitted to the Receipt Repository: <ul style="list-style-type: none">• For sales receipts: documentRequest / documentEnvelope / envelopeHash• For reports: reportRequest / reportEnvelope / envelopeHash
6.	G	Base64-encoded signature of envelope data, identical to the receipt data service XML submitted to the Receipt Repository: <ul style="list-style-type: none">• For sales receipts: documentRequest / documentEnvelope / envelopeSignature• For reports: reportRequest / reportEnvelope / envelopeSignature
4.	C	Base64-encoded serial number of the certificate used to sign the received message.

If an error code is returned, the V, D, U, H, G, and C fields must not be included in the NDEF record.



Example for Interpretation:

The example below contains fictional data.

The serial number of the receipt requested by the customer application:

NY-B999123456/98765432/0099/00004

The receipt envelope fits within the NDEF message.

The NDEF message character sequence will be:

7|R0K|V1.0|D...|U...|H...|G...|CEMTysHwssISqNnBMQi711aQ=

4.7 NAV verification code generation

The e-cash register generates a verification code for every entry submitted via the "Receipt Submission" and "Report Submission" interfaces, based on the data specified in the interface description.

Each entry submitted through the "Receipt Submission" and "Report Submission" interfaces must be assigned a recordCounter value as follows:

- The recordCounter starts from 1 at the beginning of the e-cash register's operation. The first receipt is submitted with a recordCounter value of 1.
- The recordCounter must be incremented by one for each generated receipt and report.
- There must be no gaps or repetitions in the recordCounter sequence.
- After re-personalization, the recordCounter must continue without gaps or repetitions.

The verification code must be generated using the SHA-256 algorithm. The resulting 64-character code must represent A-F hexadecimal digits in uppercase. The input character string for SHA-256 hashing consists of the following parts in the specified order:

- First 64 characters: 1-64. This is the 64-character verification code of the previous recorded entry. The previous entry is the one whose recordCounter is exactly **one less** than the current entry. If no previous entry exists (e.g., during the first entry generation after e-cash register registration), this part should consist of 64 "F" characters.
- Remaining Characters For "Receipt Submission", this consists of the value of the documentEnvelope/envelopeData field without field identifiers. For "Report Submission", this consists of the value of the reportEnvelope/envelopeData field without field identifiers.

The generated verification code must be included in the ntcaVerificationCode field of the ["Receipt Submission"](#) and ["Report reception"](#) interfaces .

The e-cash register must print the first five characters of the verification code of the previously issued receipt or report on every issued document. This must be submitted in the ntcaControlCode field of the respective entry.

If a receipt is printed in multiple copies, the same verification code must appear on all copies.

For dual-operator e-cash registers, the recordCounter and verification code must be maintained and generated separately for each taxpayer.



4.7.1 Example of NAV verification code calculation

The control code associated with the taxpayer's previous recordCounter entry:

58AB352323A4BC6DEE919345ED6FFD973C4E49BB27F964355E643812D134AD1A

Data of the current document:

```
<documentEnvelope>
  <base:envelopeData>H4sIAFtElGQC/7MJSk1OzSwocc4vSnVJLElUqMjNySu2qi
    jOtFXKKCkpsNLXLy8v1ys31ssvStc3MjAw1I/w9QlOzkjNTdT NzCsuScx
    LTIWy41JQsMINLUkEGQHiALkp+cmlual5JX6luUmpRXZRhibmBkBgp
    pA80w1gdxDS00UdTBDJGH2GOjT6a6+wA+FPs6q8AAAA=</base:envelopeData>
  <base:envelopeHash>cryptoType="SHA-256">D110A234391627E0C202AD3E1C855CC66452C0C732006DE5989F3C1AE1
  9E57AC</base:envelopeHash>
  <base:envelopeSignature>MEUCIQCqCATvvP3yvf6cOSoKRGLSjPVXFpPacd4kt/o
  HWkonXgIgbwzORu3gNaw/qjsG5liOMVXMfS8At8tCpRhfHfgXJuU=</base:envelopeSignature>
</documentEnvelope>
```

In the example, the data in bold should be concatenated:

58AB352323A4BC6DEE919345ED6FFD973C4E49BB27F964355E643812D134AD1AH4sIAFtElGQC/7MJSk1OzSwocc4vSnVJLElUqMjNySu2qijOtFXKKCkpsNLXLy8v1ys31ssvStc3MjAw1I/w9QlOzkjNTdT NzCsuScxLTIWy41JQsMINLUkEGQHiALkp+cmlual5JX6luUmpRXZRhibmBkBgpA80w1gdxDS00UdTBDJGH2GOjT6a6+wA+FPs6q8AAAA=

The hash value of this, i.e., the NAV control code:

8D982E09861A1B73B2D164C556D11AEF99BC519D31A479AA25ABCFA6E12D2EF

4.8 Technical description of the services

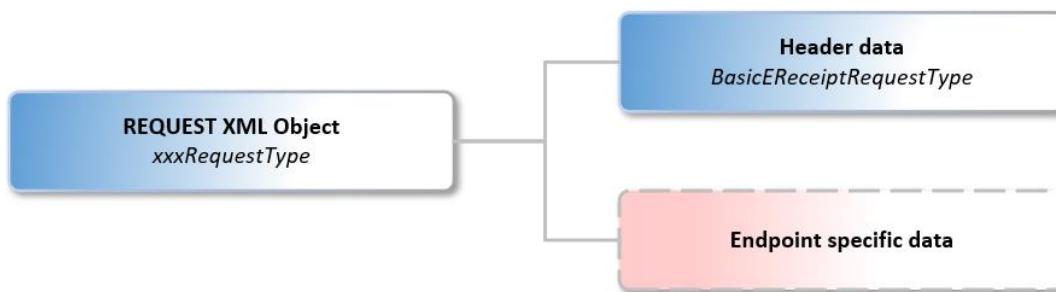
4.8.1 General technical data

During communication between the e-cash register and the NAV-I system, every message must consist of a character sequence encoded according to the UTF-8 character set. The e-cash register must be capable of storing the public key of the NAV SSL certificate and may establish a connection only after verifying this certificate. Messages must be sent to the central system over an SSL channel using the HTTP/POST or HTTP/GET method. By default, the applicable method is HTTP/POST, and any services available through a different method will be explicitly indicated in their respective service descriptions.

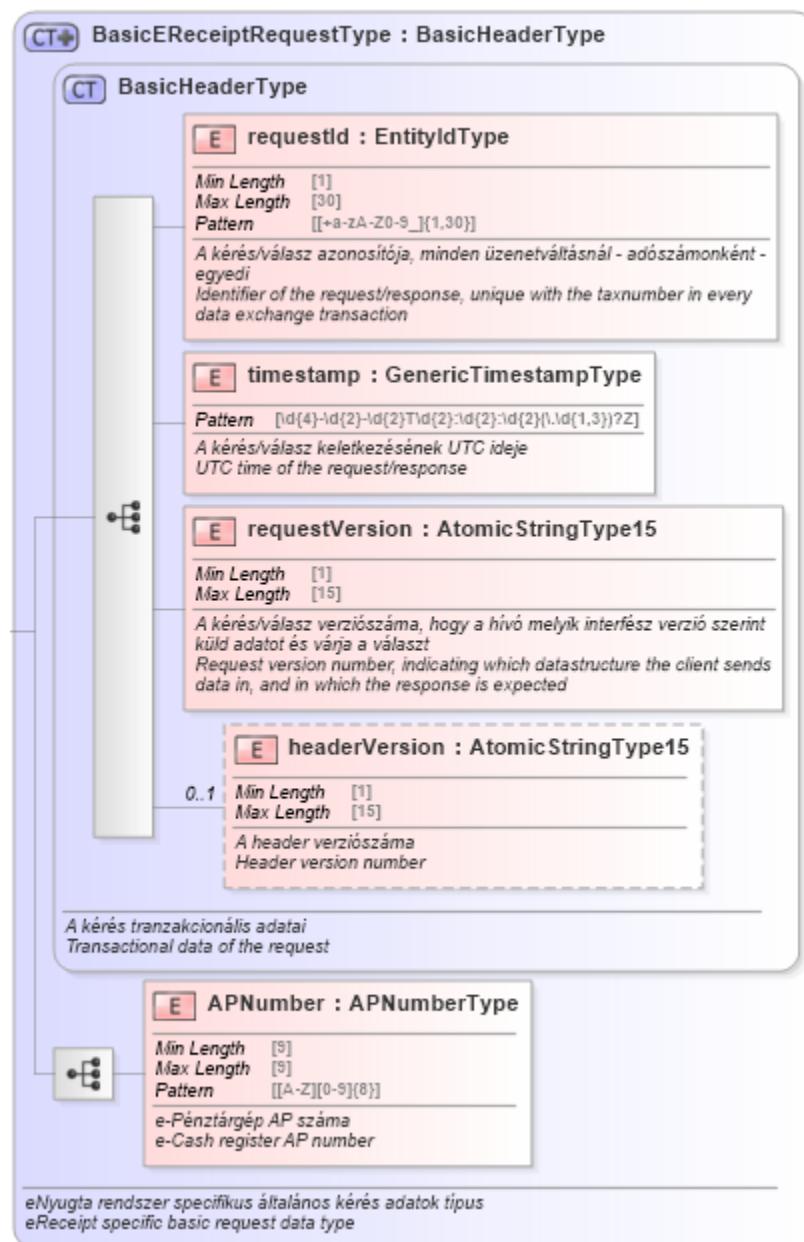
For service endpoints accessible via the HTTP POST method, the appropriate XML request must be sent in the body, and the server will return an XML response in the response body.

The caller defines the requested operation by addressing the appropriate endpoint and structuring the XML request accordingly. The server will always respond with an XML message and a corresponding HTTP response code based on the validity of the request.

Every request (XML structure) follows the unified format outlined below:

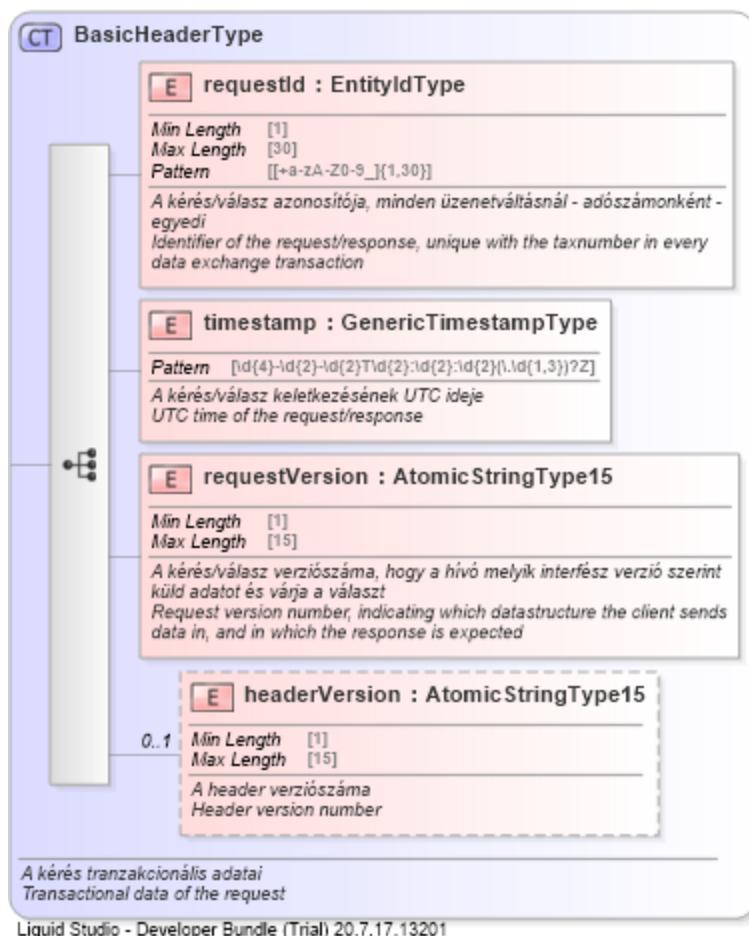


The structure of the header data sent by the e-cash registers is shown in the following diagram:



- **requestId** – the unique identifier of the request and its corresponding response, assigned by the caller. The identifier must not be repeated from an AE (an AP number); if duplicated, the server will reject the request.
- **timestamp** the timestamp of when the request was generated. If the AE's clock deviates by more than one hour from the server's clock, the server will reject the request.
- **requestVersion** – the interface version number, which defines the request data structure.
- **headerVersion** – the XML header version number.
- **APNumber** – the AP number of the e-cash register submitting the request.

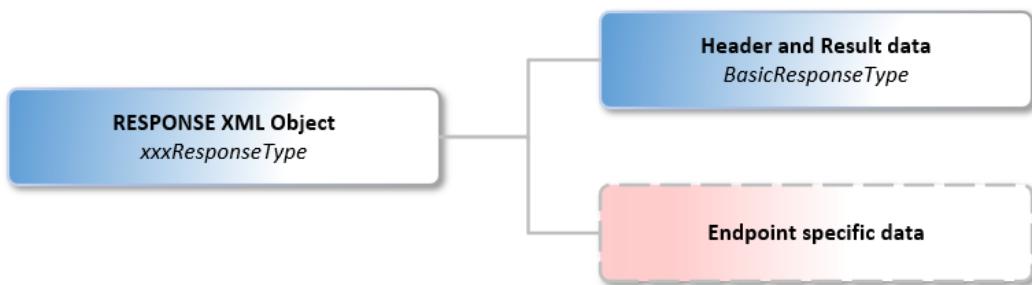
The structure of the header data sent by customer applications is shown in the following diagram:



The interpretation of the fields is the same as those listed in the request header of the e-cash register interface.

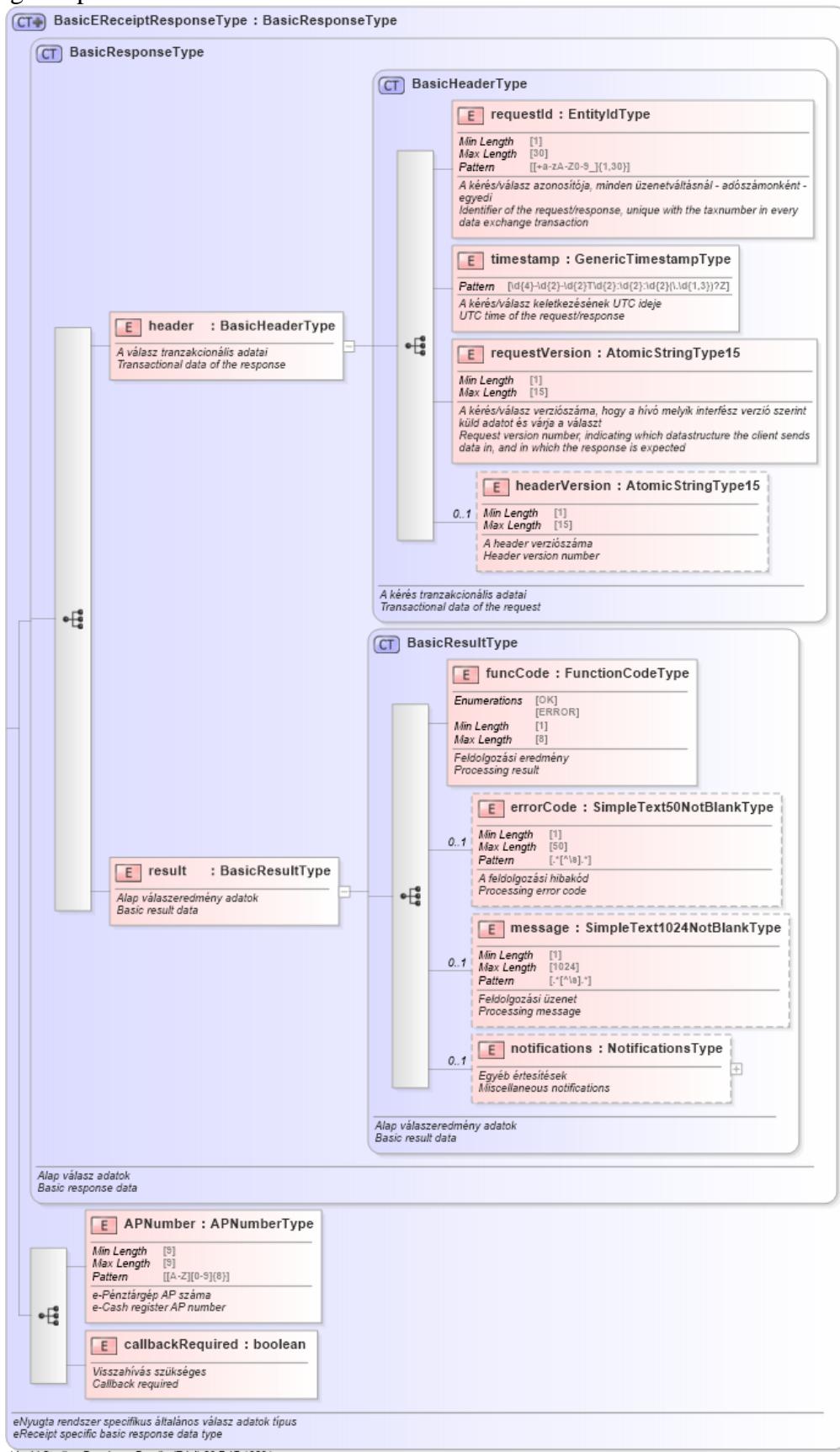


The response XML structure consists of the following main components:





The structure of responses received on the interface called by e-cash registers consists of the following components:

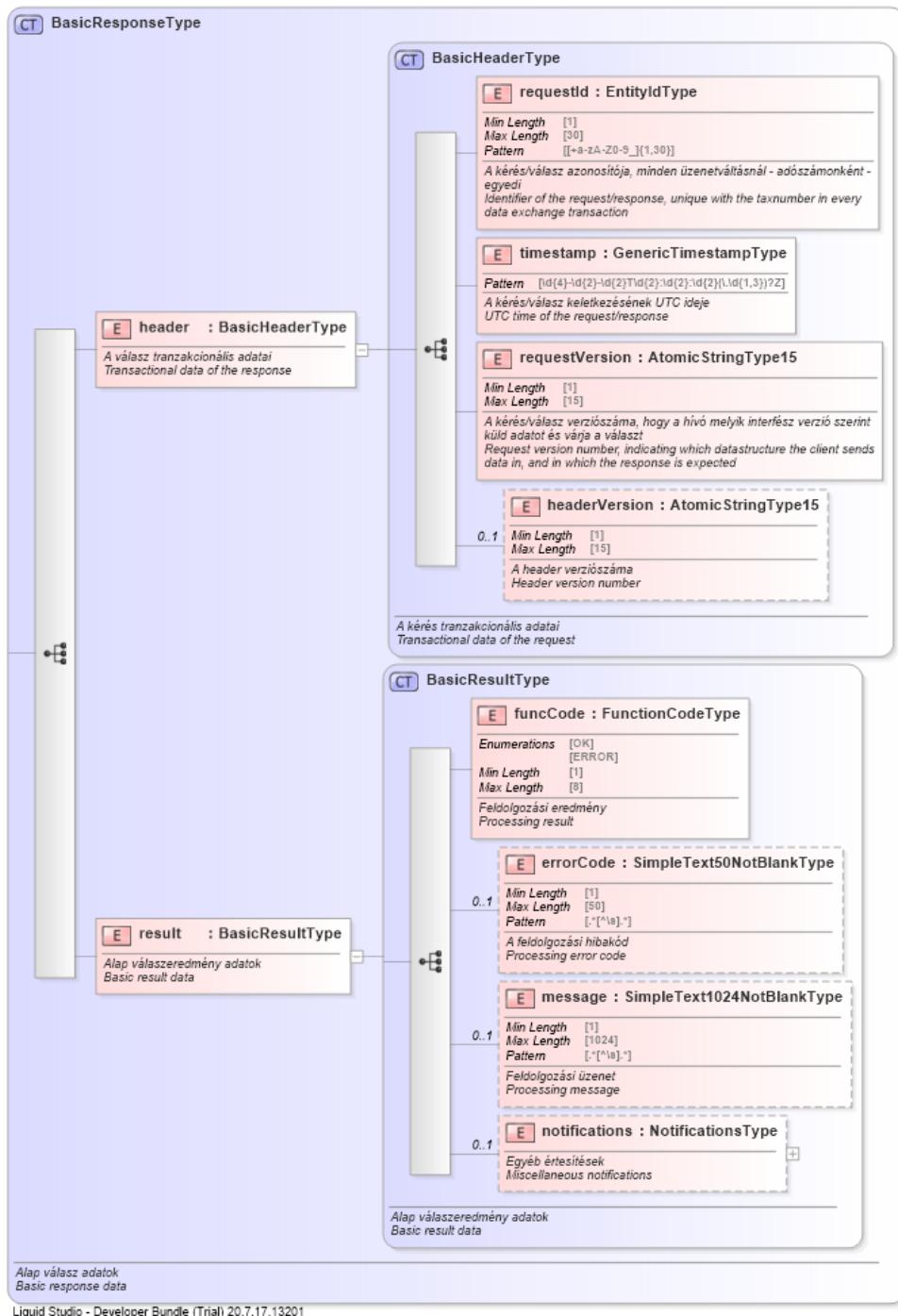


Liquid Studio - Developer Bundle (Trial) 20.7.17.13201



- **header** – header data:
 - **requestId** – the unique identifier of the request to which the server sends the response.
 - **timestamp** – the timestamp of the response according to the server's clock.
 - **requestVersion** – the version number of the interface, which determines the response data structure.
 - **headerVersion** – the version number of the XML header.
- **result** – result data:
 - **funcCode** – a code indicating the result of the server-side processing of the request, "OK" for successful processing, "ERROR" for an error response.
 - **errorCode** – an optional field that returns an error code in case of an error, consisting of a letter and four digits, e.g., "E0123."
 - **message** – an optional field containing a textual message indicating the result of the processing. In the e-receipt system, it includes textual explanations of error codes.
 - **notifications** – an optional field containing other notifications from the server. Not used in the e-receipt system.
- **APNumber** – the AP number present in the request being responded to.
- **callbackRequired** – a boolean flag indicating to the e-cash register that after processing the response, it must call the Communication Manager service. Further details can be found in the "[Communication Manager](#)" subsection.

The response XML header for the Receipt Repository interface called by customer applications:



The interpretation of the fields is the same as those listed in the e-cash register interface response header.

The e-cash register must be able to establish a connection with NAV-I, send and receive messages, and execute instructions received from NAV-I. Detailed descriptions and specifications of the individual interfaces can be found in the chapter "[Business Services Provided by NAV for E-Cash Registers.](#)"



The service endpoint context roots can be one of the following:

- **/eReceipt/v1** – endpoints for receipt and report submission called by e-cash registers
- **/eReceiptMgmt/v1** – control endpoints for e-cash registers
- **/eDocumentStore/v1** – endpoints accessible by the customer application in the receipt repository

The technical description of the service endpoints introduced in the chapter "[Business Services Provided by NAV for E-Cash registers.](#)" contains the context root of the respective endpoint.

If the connection attempt initiated by the e-cash register fails, the communication can be retried at the earliest five seconds after the previous attempt, up to a maximum of two additional times. If the connection between the e-cash register and the NAV-I central system is restored, the offline-issued receipts must be transmitted to the central system. When submitting, the most recently generated receipts must be sent first (LIFO principle – Last In, First Out).

If these attempts remain unsuccessful, the AE may attempt to reconnect with the NAV server 30 minutes after the last failed call.

4.8.2 HTTP headers

The services presented in the chapter "[Business Services Provided by NAV for E-Cash Registers.](#)" must always be called using the HTTP POST method. If a service needs to be used differently, it will be specified in its description. For every service called with the POST method, the following HTTP header fields must be provided:

- content-type=application/xml ; charset=UTF-8
- accept=application/xml

4.8.3 HTTP status codes

A correct request will always return an HTTP 200 response. However, this does not necessarily indicate that the business execution of the formulated request was successful, only that the request was technically well-formed, and the called resource was able to read and accept it. This means that the request conformed to the technical description of the service, and the message structure matched the schema (XSD) describing the service call. The correctness of the business data sent in the XML, and its compliance with the business object, must be ensured by the e-cash register during message composition, using an XSD validation on the XML.

For results returned due to incorrect requests or other technical errors, refer to the error code table in the "[Error Handling](#)" chapter.

4.8.4 Response time, timeout

The server typically responds within **500 ms**. The blocking timeout value for synchronous calls is **5000 ms**. On the client side, response times exceeding this value can be treated as a timeout.

4.8.5 Converting local time to UTC

The NAV-I receiving server obtains time settings from a closed NTP server inaccessible to the external world. In the case of hardware-based e-cash registers, the system time must be synchronized with the NTP server provided by the mobile service provider. Cloud-based e-cash registers must also synchronize their system time. Time synchronization can be performed via: <http://www.pool.ntp.org/zone/hu> (an NTP client is required for connection).

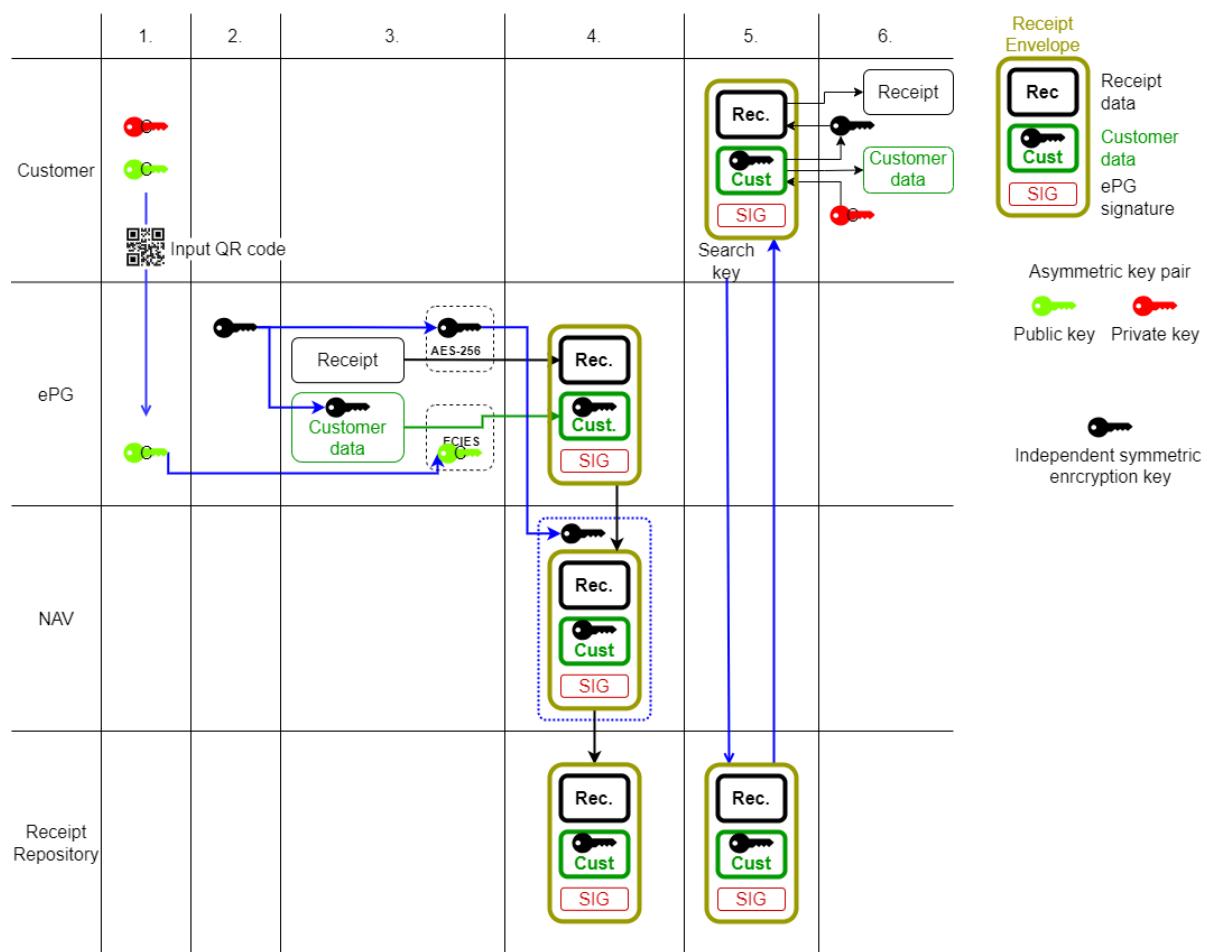
If a time field is included in a message sent during communication, the system time must be provided in UTC format.

4.8.6 Radius server

For hardware-based e-cash registers, before establishing a connection, the mobile service provider must register the device's network-registered IP address in the NAV-I system via the RADIUS protocol.

4.9 Receipt issuance, submission, and querying technological process

4.9.1 When using a customer application

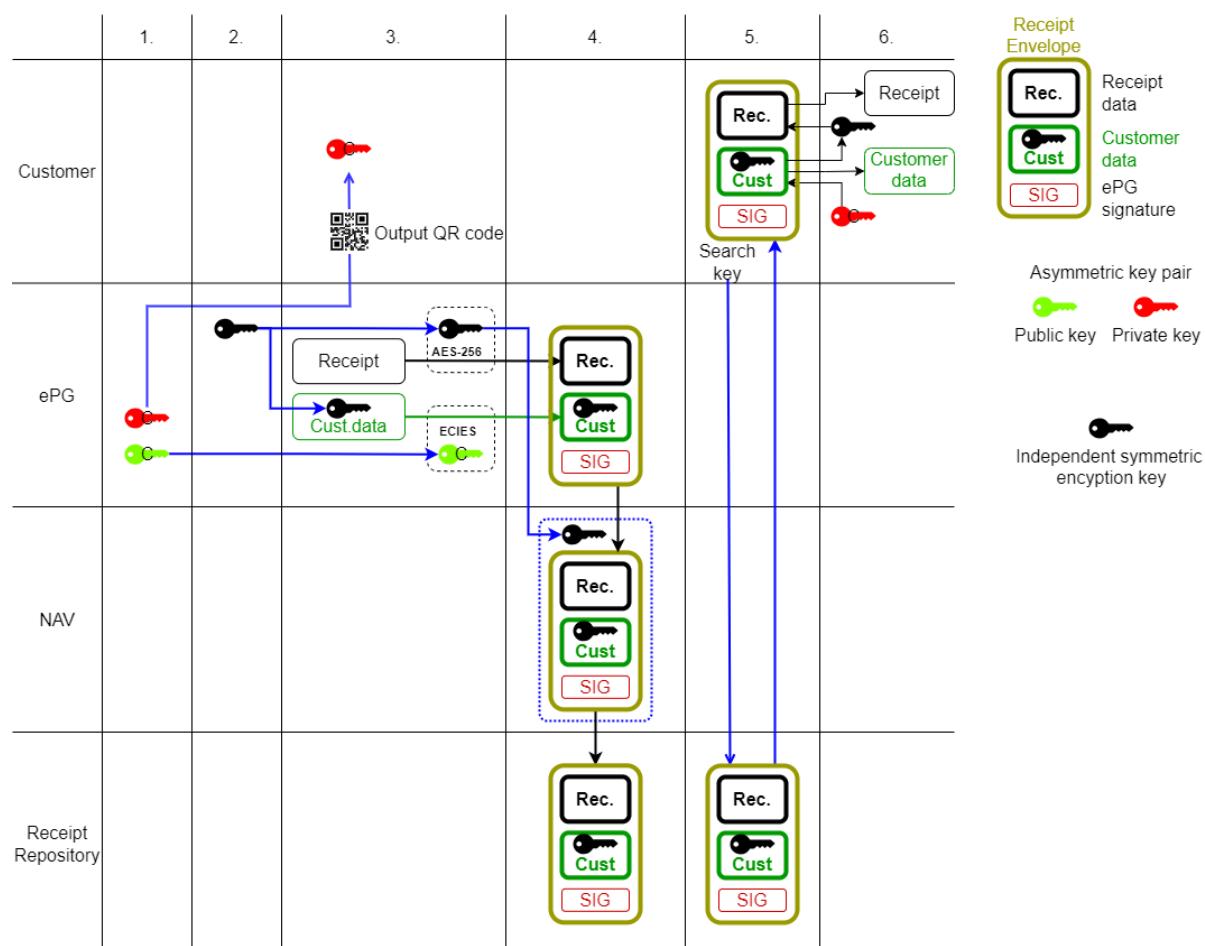


1. Before making a purchase, the customer application generates an asymmetric encryption key pair (private and public keys) and a QR code. The generated public key (marked as "V" green key in the diagram) and the QR code creation timestamp are embedded into the QR code. More details on QR code generation are provided in the section "[Generating input QR code by E-cash register](#)" section.
2. The e-cash register generates a 256-bit AES symmetric encryption key to encrypt the receipt data
3. The e-cash register prepares and encrypts data. The receipt data is encrypted using the AES-256 algorithm with the independent symmetric key. The customer data includes the

independent symmetric key and any optional customer data. The customer data is encrypted using the customer's public key with the ECIES encryption algorithm. More details on encryption are available in the "[Encryption](#)" section.

4. The receipt and customer data are digitally signed using the e-cash register's private signing certificate. More details on digital signing are found in the "[Creating Signatures](#)". The complete receipt envelope is sent to NAV-I as described in the "[Receipt Submission](#)" subsection.
5. Once successfully sent, the e-cash register provides visual and audio notification. The customer can now retrieve the receipt from the receipt repository in their application using the search key.
6. The customer downloads the receipt envelope in their application. The customer's private key (marked as "V" red key in the diagram) is used to decrypt the customer-specific data. The independent symmetric key (black key in the diagram) is then used to decrypt the receipt data.

4.9.2 Without using a customer application





If the customer does not have a customer application, the process described in the previous section changes as follows:

1. The e-cash register generates a unique asymmetric encryption key pair for the customer for the given receipt. These keys are marked as "V" (green and red keys) in the diagram.
2. After payment, the e-cash register generates the digital receipt and, if requested by the customer, prints a copy. The e-cash register prints a QR code on the receipt copy, which is generated using the private key created in step 1 (marked as "V" red key) and the search key. If the customer does not request a paper copy, the QR code must be displayed on the customer screen for scanning with the customer application. More details on QR code generation are provided in the section "[Formation of the output QR code for the E-cash register](#)". Using the QR code printed on the receipt copy or displayed on the customer screen, the installed customer application can download, decrypt, and display the receipt envelope and its contents

4.10 Search key

Both the e-cash register and the customer application must be able to generate a unique encryption key pair for each purchase.

The **search key** is the SHA-256 checksum of the public key of the customer's encryption key pair (raw, compressed).

Receipts can be queried from the receipt repository using the search key and the search date (either from the customer QR code or the receipt issuance date).

The e-cash register submits the search key value in the **searchKey** field.

5 Business services provided by NAV for E-Cash registers

5.1 Device registration

According to Annex 2, Part A, Section 14. b) of the Regulation, the e-cash register must perform device registration in collaboration with the NAV infrastructure.

The e-cash register must complete the device registration by invoking this service, which is used for setting up a new e-cash register in the system and putting it into operation.

5.1.1 Business description of the service

The device registration service is used for setting up a new e-cash register system and putting it into operation. During the registration process, the e-cash register receives all the necessary data required for its operation and is assigned to the taxpayer operating the e-cash register.

Every e-cash register must use this service before starting its operation. Until the registration is successfully completed, the e-cash register cannot be used.

Prerequisites for invoking the service:

- The e-cash register must have an AP number, which is assigned by the e-cash register distributor.



- For hardware-based e-cash registers, a standard X.509 format authentication and signing certificate request (Certificate Signing Request – CSR) must be generated. The CSR does not need to contain business data, as the NAV-I system will insert these into the certificate.
- For cloud-based e-cash registers, the authentication certificate is used by the central application module handling NAV data communication. There is no need for separate authentication certificates per AP number.
- The e-cash register must have a valid installation code, which the taxpayer operating the e-cash register can request from NAV in a specified manner. Entering the installation code in the e-cash register must be possible at least via the e-cash register keyboard (QR code scanning is also allowed).

If the device registration service returns an error response, the registration fails, and the service can be called again.

Upon a successful request, the registration service response contains two URLs, from which the authentication and signing certificates can be downloaded. Before the first download attempt, the e-cash register must wait 5 seconds. If the certificate is not yet available, at least 10 seconds must pass between retry attempts.

If 5 minutes pass after receiving the URLs without a successful certificate download, the registration is considered unsuccessful, and the service must be called again. For reattempting registration, new CSRs with newly generated keys must be submitted.

After successfully downloading the certificates, the e-cash register must invoke the NAV Hello service. Until the Hello service is successfully called, the device registration can still be repeated. Once the Hello service has been successfully invoked, the registration is considered complete, and no further registration attempts can be made.

After successful registration, the e-cash register is ready for use.

If an online cash register model compliant with 48/2013 (XI. 15.) NGM regulation has been converted into an e-cash register through software updates, the newly manufactured e-cash register with the updated software must be registered in the system using this service.

For a successful registration, the distributor must perform a preliminary registration in the NAV system via the ePG Portal, where the basic technical data of the e-cash register (e.g., IMEI, IMSI) must be provided.

5.1.2 Technical description of the service

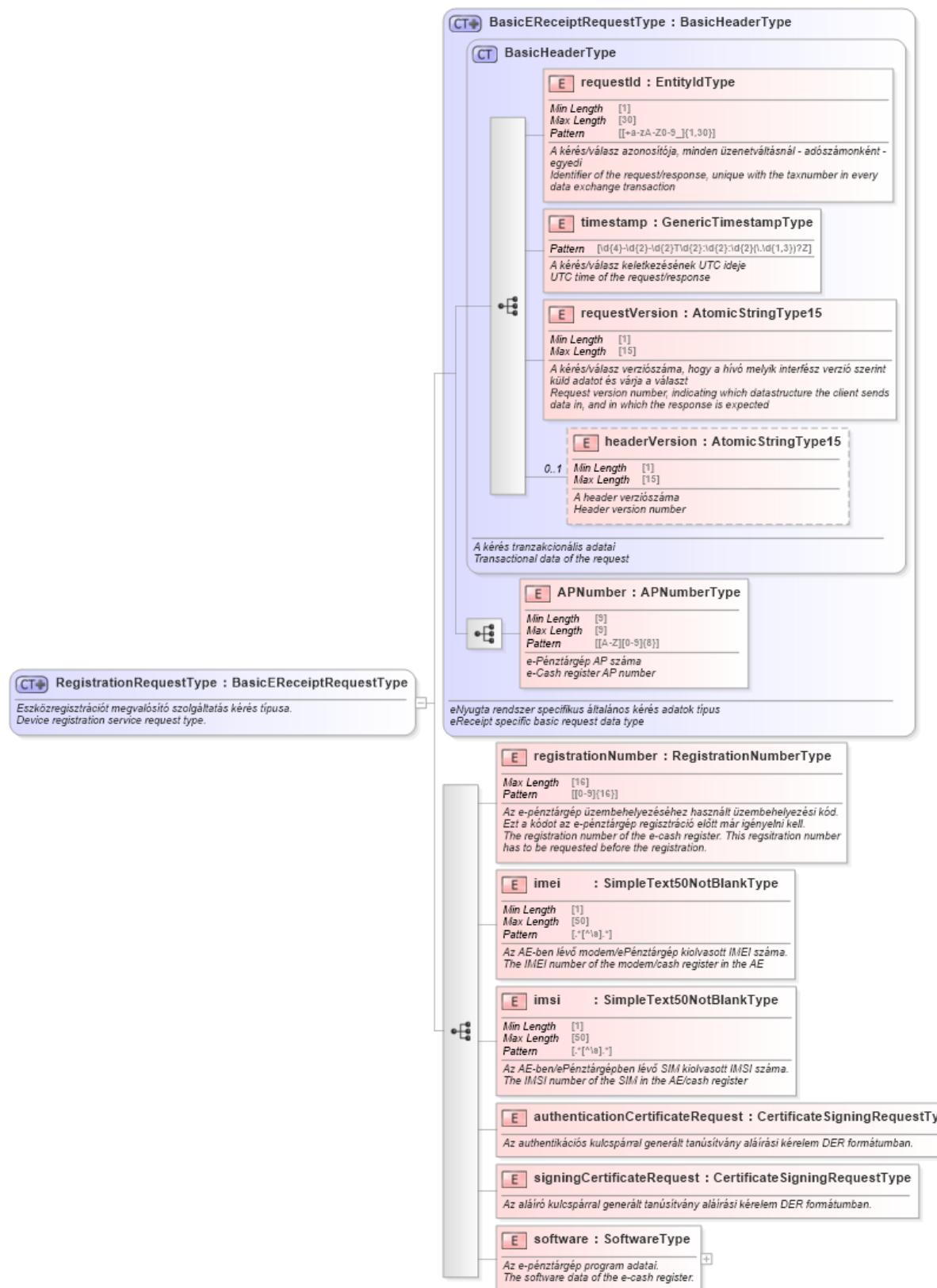
Device registration is executed via the "registration" service.

- Context root: /eReceiptMgmt/v1
- URL: /registration
- Request Object: RegistrationRequest. Technical details for this object are described in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- Response Object: RegistrationResponse. Technical details for this object are described in the "[Description of business data content \(XSD model types and elements\)](#)" section.

This service can be called without using an authentication key.



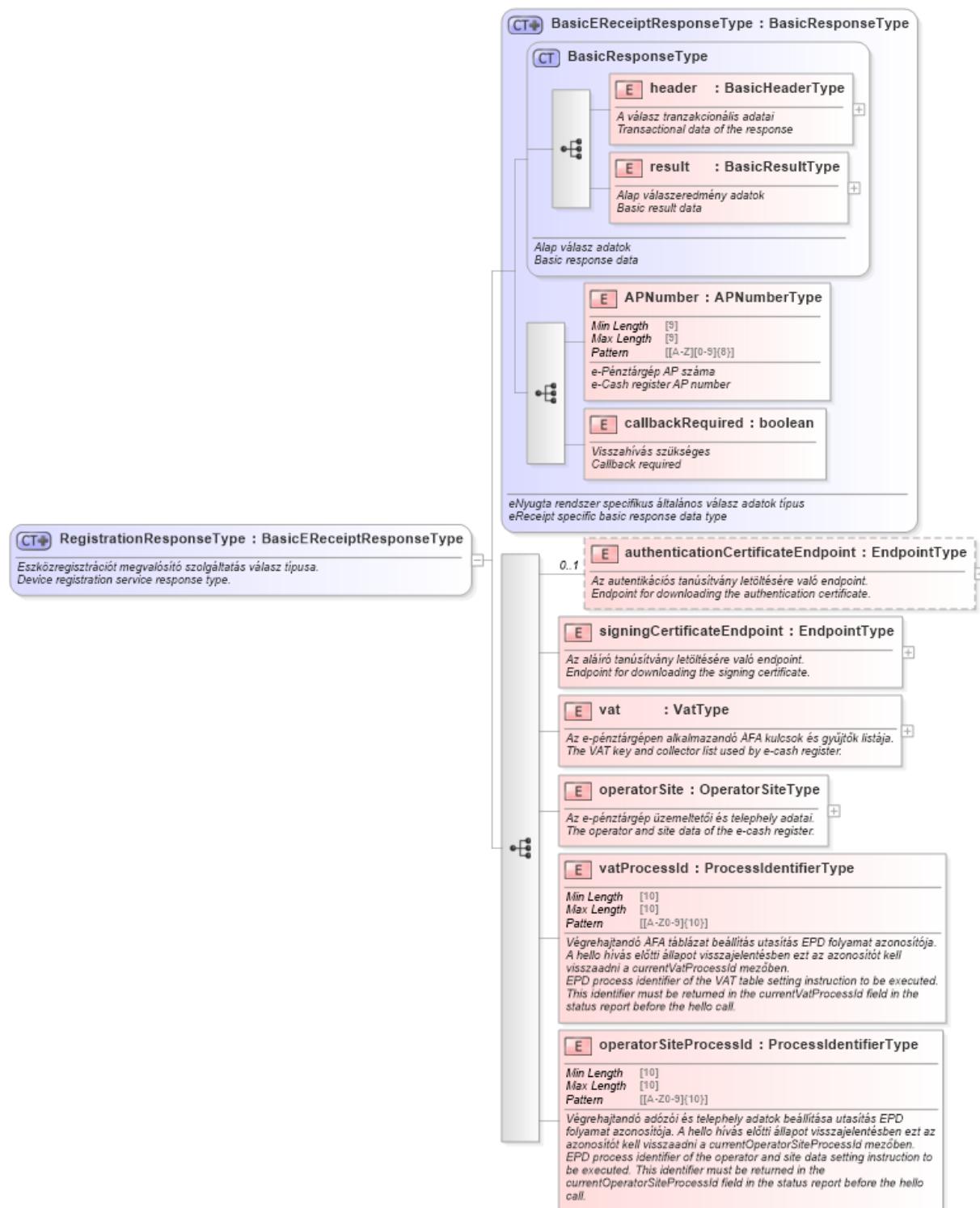
RegistrationRequest request object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201



RegistrationResponse response object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

5.2 Receipt submission

According to Annex 2, Part A, Section 5 of the Regulation, the e-cash register must transmit the required receipts to the e-receipt archive (Nyugtatár) and fulfill its data reporting obligations to NAV-I. For receipts such as E-receipt (e-nyugta) and its modifications or cancellations, Simplified invoice and its modifications or cancellations, Invoice and its modifications or



cancellations, Fuel card sales receipt, Health card sales receipt, Hotel charge transfer receipt, Consumption summary receipt, The e-cash register must immediately send the receipt to NAV-I via a machine interface after each issuance. For dual-operator e-cash registers, each taxpayer must separately call this interface.

5.2.1 Business description of the service

This service allows the e-cash register to submit receipts to NAV in compliance with legal regulations:

- E-receipt (e-nyugta) and its modifications or cancellations
- Simplified invoice and its modifications or cancellations
- Invoice and its modifications or cancellations
- Fuel card sales receipt
- Health card sales receipt
- Hotel charge transfer receipt
- Consumption summary receipt

The e-cash register must immediately send the receipt to NAV-I via a machine interface after each issuance.

For dual-operator e-cash registers, each taxpayer must separately call this interface.

Handling Network Outages According to Annex 2, Part A, Section 17. b) of the Regulation, If the network is unavailable, the receipt submission must be postponed. As soon as connectivity is restored, all pending receipts must be sent immediately. Delayed submission must be flagged accordingly. Receipts must be submitted in reverse order (starting with the most recently issued). If a new receipt is generated during this process, it must be sent first.

Receipt Data Structure Each submitted receipt consists of three parts:

- Receipt Data - Sent to both NAV and the e-receipt archive (Nyugtatár). - The buyer can access and view this data from Nyugtatár.
- Customer Attachment - Additional buyer-seller data not part of the receipt. - Stored in the Nyugtatár. - Sent encrypted, and only the buyer can decrypt it.
- Receipt-Related Data Reporting - Additional information such as changes in the cash register drawer when an invoice is issued. - Included in the receipt envelope and accessible to the buyer.

Receipt Envelope Submission:

- The receipt and its related reporting data are encrypted and sent to NAV and the buyer.
- Buyer-specific data is sent in encrypted form so that only the buyer can access it.

The e-cash register side integrity checking solution specified in Section 23 (3) of the Decree must verify the data to be submitted from syntactic and semantic aspects before generating the electronic receipt, the minimum element of which is the XSD validation of the XML data to be submitted. If errors are detected, they must be corrected before submission. If the prepared data package contains errors, it cannot be used to generate an e-receipt, cannot be submitted to NAV, and prevents transaction completion (sale, payment, or any economic event).

Search Key Submission The search key must be included with the receipt. If the buyer provided a search key before the transaction, that key must be sent. If no search key was provided, the e-cash register must generate a unique key per receipt.

Receipt Envelope Encryption & Signing The receipt envelope is sent in an encrypted and signed format. Signing details can be found in the “[Creating signatures](#)” section. Encryption details are explained in the [“Encryption”](#) section.

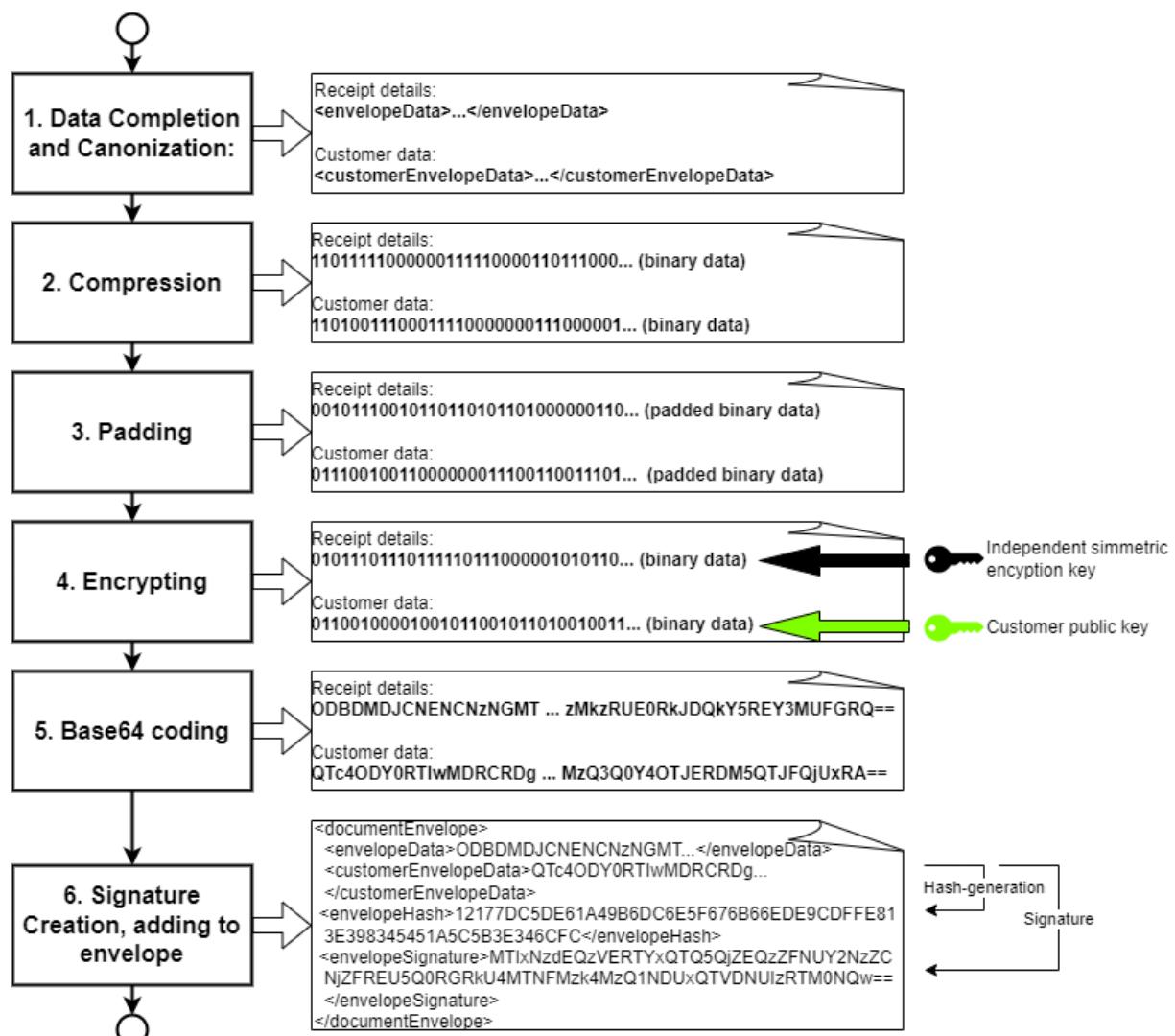
Visual & Audio Confirmation of Successful Submission After a successful submission, the e-cash register must provide visual and audio feedback: Visual Feedback: A GIF image (potentially animated) is displayed. Audio Feedback. A WAV sound file is played. These files are published by NAV and can be downloaded from the official website.

Receipt Envelope Compilation The data sent to NAV must be included in the **documentEnvelope** field of the receipt submission request. The following diagram illustrates the receipt envelope compilation process.

5.2.2 Receipt envelope compilation

The data to be sent to NAV must be included in the **documentEnvelope** field of the receipt submission request.

The process of compiling the envelope is illustrated in the following diagram:





The steps for compiling the DocumentEnvelope and CustomerEnvelope:

1. Data Completion and Canonization: First, the appropriate data structure must be filled in (the description of data structures can be found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.
 - For e-receipts and their modifications or cancellations:
 - Receipt data (EnvelopeData):
 - Receipt: ReceiptCore
 - Customer Data (CustomerEnvelopeData):
 - Independent symmetric encryption key
 - Receipt attachment: ReceiptAdditional
 - For simplified invoices and their modifications or cancellations, as well as invoices and their modifications or cancellations:
 - Receipt Data (EnvelopeData):
 - Receipt Data: SimplifiedInvoiceCore (The SimplifiedInvoiceCore simplifiedInvoice section must be submitted according to the online invoice structure, described in the NAV Online Invoice System's "Invoice Data Service REST API Interface Description and Developer Documentation" version 3.0 under the "Structure of the InvoiceDataType Complex Type" chapter.)
 - Invoice data service: SimplifiedInvoiceControl
 - Customer Data (CustomerEnvelopeData):
 - Independent symmetric encryption key
 - Receipt attachment: SimplifiedInvoiceAdditional
 - For fuel card sales receipts, health card sales receipts, hotel recharges, or consumption summary receipts:
 - Receipt Data (EnvelopeData):
 - Receipt: OtherDocumentCore
 - Customer Data (CustomerEnvelopeData):
 - Independent symmetric encryption key
 - Receipt attachment: OtherDocumentAdditional

The completed data structures must be converted to a canonical form (<https://www.w3.org/TR/xml-c14n11/>) and validated according to the NAV-issued methodology. If an error is detected, it must be corrected before submission.

2. Compression: The data generated in the previous step must be compressed according to the "[Compression](#)" chapter. The result is binary data can be found in "[Compression](#)" chapter.
3. Padding: The binary data from the compression process must be prepared using **PKCS#7 padding** for the next step.
4. Encryption: Receipt data (EnvelopeData) must be encrypted using the independent symmetric key and customer data must be encrypted using the customer's asymmetric public key according to the method described in the "[Encryption](#)" chapter. The result of encryption is binary data.
5. Base64 encoding: The binary data generated in the previous step must be Base64 encoded.
6. Signature Creation: The Base64-encoded receipt data from the previous step must be placed in the envelopeData field of the DocumentEnvelope, he customer data package must be placed in the customerEnvelopeData field. From this, **envelopeHash** and **envelopeSignature** must be generated according to the "[Creating Signatures](#)" chapter.



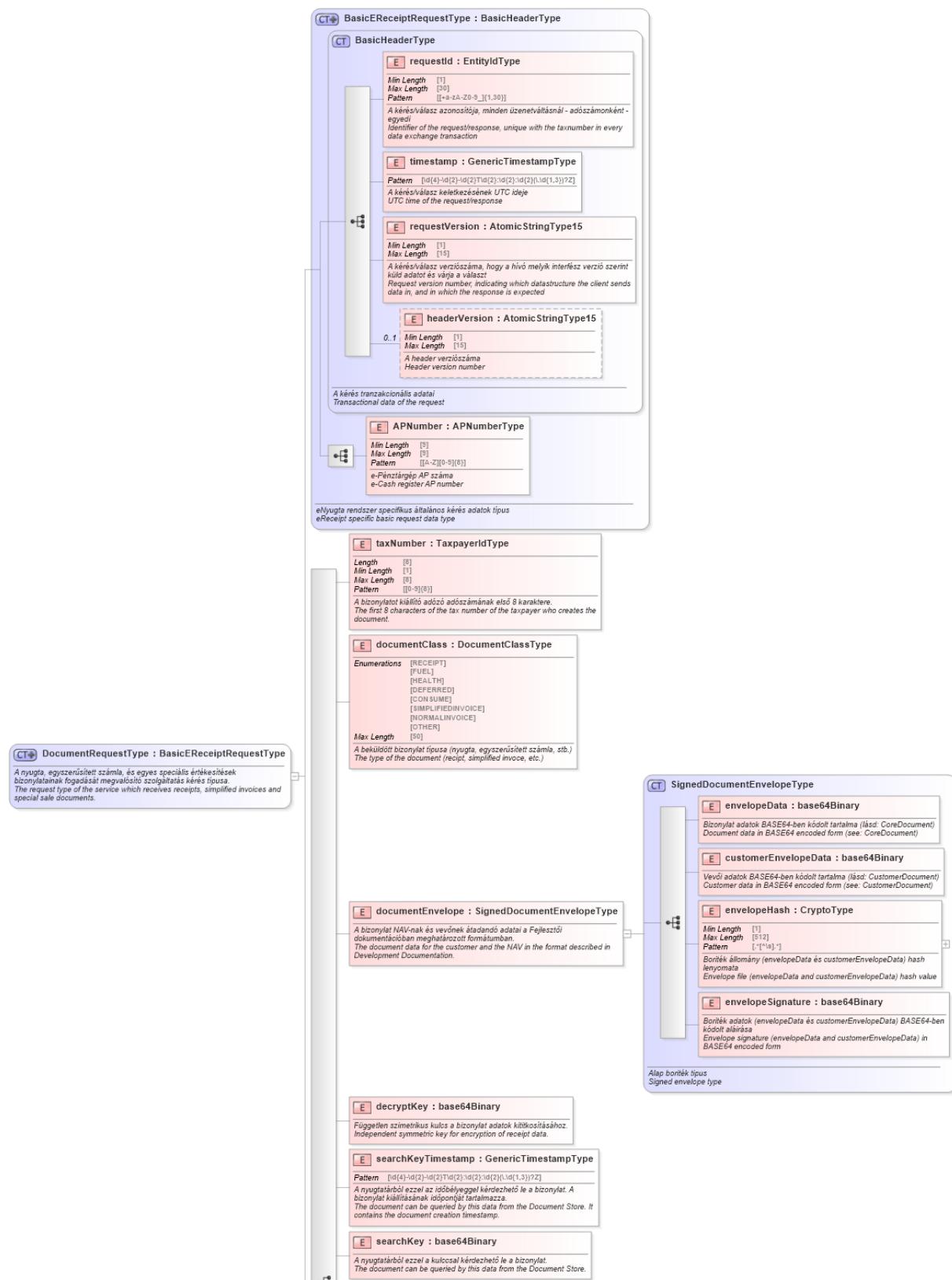
5.2.3 Technical description of the service

The receipt submission is implemented by the "document" service.

- Context root: /eReceipt/v1
- URL: /document
- Request Object: DocumentRequest. The technical description of the service request object is found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.
- Response Object: DocumentResponse. The technical description of the service response object is found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.



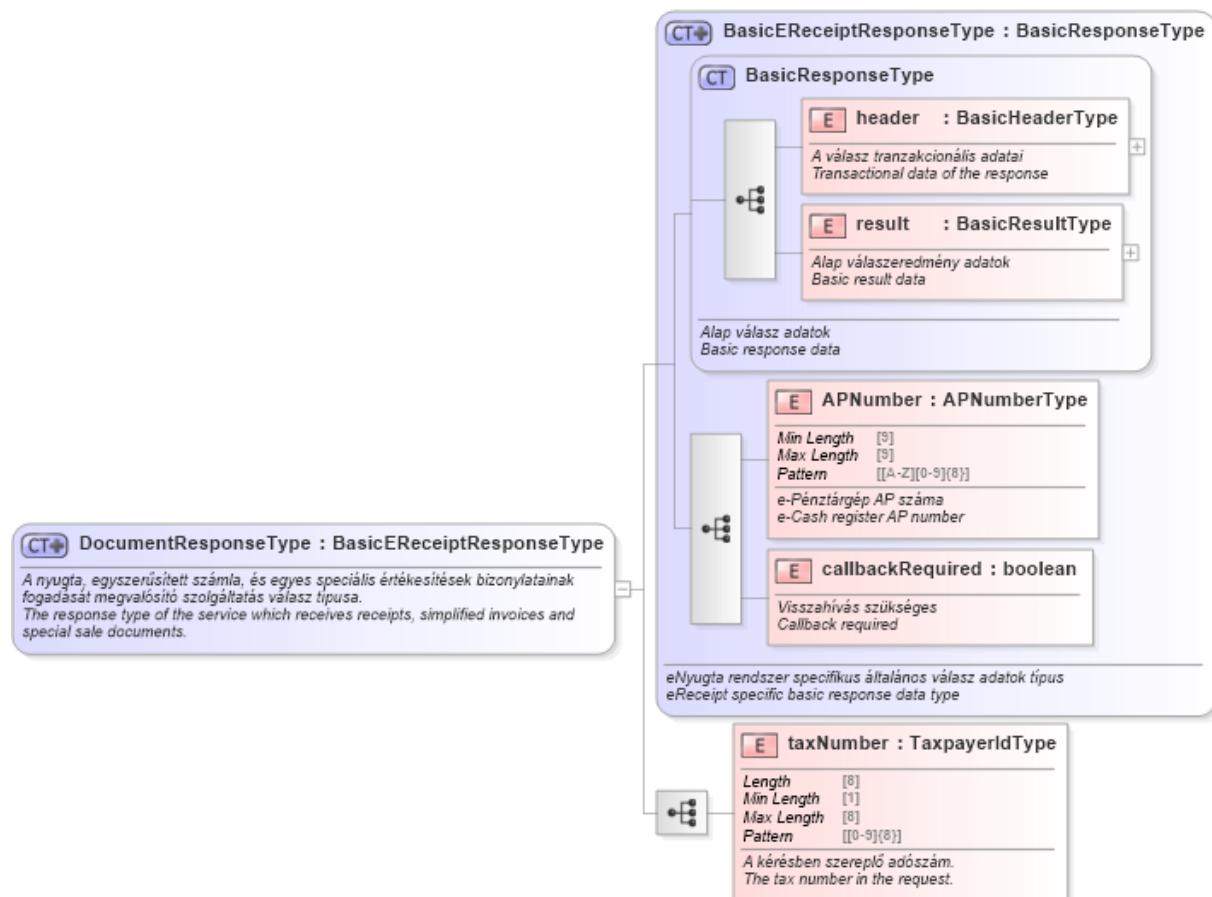
DocumentRequest request object:





E qRCodeExpired : boolean
A QR generálása a fejlesztő dokumentációban meghatározott időintervallumnál régebben készült.
E offlineCreated : boolean
A hírközlő hálózat elérhetetlenségekor megszűnés esetén az érintett bizonylatokat és adatszolgáltatásokat haladéktalanul el kell külni, a késleltetett küldés ténylet ennek a mezeinek a kitöltésével kell jelezni. Whether the document was created offline. When the communication network is online again, the document has to be sent immediately, with this flag.
E cashRegisterSignCertificate : CertificateType
Az e-pénztárgép aláíró tanúsítvány DER formátumban. A tanúsítvány privát kulcsával kerülnek elárasztásra az e-pénztárgép által küldött üzeneteik. Az aláíró szaccal kapcsolatos követelmények a Fejlesztői dokumentáció „Aláíró képzése” fejezetében. The sign certificate of the e-cash register in DER format. The private key of this certificate has to be used to sign the messages of the e-cash register. The requirements of the sign key can be found in the "Signing" chapter of the Development Documentation.
E recordCounter : nonNegativeInteger
Az e-pénztárgépen utoljára kiállított bizonylat recordCounter értéke. Leírását lásd a Fejlesztői dokumentáció „NAV ellenőrző kód képzése” fejezetében. The recordCounter value of the actual document. See further information in the "Generate NAV verification code" chapter of the Development Documentation.
E lastRecordCounter : nonNegativeInteger
Az e-pénztárgépen utoljára kiállított bizonylat recordCounter értéke. Leírását lásd a Fejlesztői dokumentáció „NAV ellenőrző kód képzése” fejezetében. The recordCounter value of the last document in the e-cash register. See further information in the "Generate NAV verification code" chapter of the Development Documentation.
E ntcaVerificationCode : SHA256Type
Min Length [1] Max Length [64] Pattern [[0-9A-F][64]] NAV ellenőrző kód. Leírását lásd a Fejlesztői dokumentáció „NAV ellenőrző kód képzése” fejezetében. Ntca verification code. See further information in the "Generate NAV verification code" chapter of the Development Documentation.
E documentEnvelopeVersion : AtomicStringType15
Min Length [1] Max Length [15] 0..1 Amennyiben a beküldött boríték tartalmát egy régebbi interfejsz verzió szerint állította elő a pénztárgép (újratöltés esetén), mint a jelenlegi interfejsz verzio (requestVersion), akkor annak verziójának itt meg kell megadni. If the contents of the submitted envelope were produced by the e-cash register according to an older interface version (in case of resending) than the current interface version (requestVersion), then its version number must be entered here.
E sendMissingDocumentProcessId : ProcessIdentifierType
Min Length [10] Max Length [10] Pattern [[A-Z0-9][10]] 0..1 Amennyiben a bizonylat beküldése a SEND_MISSING_DOCUMENT felszólítás végrehajtásához köthető, akkor meg kell az abban kapott folyamat azonosítót. Normál beküldés esetén nem kell megadni. If the document is sent due to the execution of the SEND_MISSING_DOCUMENT command, the process ID received in it must be entered here. It is not required for normal submission.

DocumentResponse response object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

5.3 Report reception

According to Annex 2, Part A, Section 5 of the Regulation, the e-cash register must transmit the required receipts to the receipt repository and fulfill data reporting obligations to NAV. For cash register opening (day opening) receipts, daily sales reports, cash register reports, cash flow receipts, and all other receipts, this obligation must be fulfilled by calling this service from the e-cash register.

5.3.1 Business description of the service

This service receives the following types of report receipts issued by the e-cash register:

- Cash Register Opening Receipt
- Daily Sales Report
- Cash Register Report
- Cash Flow Receipt
- Other Reports

The e-cash register must send each receipt individually to NAV-I via a machine interface immediately after issuance. In the case of a dual-business e-cash register, each taxpayer must invoke this interface separately for their respective receipts.



According to Annex 2, Part A, Section 17.b of the Regulation, if a network communication outage occurs, the affected receipts and data reports must be sent immediately after the outage is resolved, marking the delayed transmission as specified in the Developer Documentation. In such cases, offline-issued receipts must be submitted in reverse chronological order, starting with the most recently issued receipt. If a new receipt is issued while submitting past receipts, the newly issued receipt must be submitted first.

The e-cash register side integrity checking solution specified in Section 23 (3) of the Decree must verify the data to be submitted from syntactic and semantic aspects before generating the electronic receipt, the minimum element of which is the XSD validation of the XML data to be submitted. If validation errors are detected, they must be corrected before submission. If there is an error in the prepared data package, the report cannot be generated, cannot be submitted to NAV, and the transaction cannot be completed on the e-cash register. In such cases, manual receipt issuance or switching to another e-cash register solution may be necessary.

If a cash flow receipt or another non-fiscal receipt involves a customer (e.g., a deposit or prepayment receipt that must be provided to the customer), the search key and its timestamp must be submitted, allowing the receipt to be retrieved from the receipt repository.

- If the customer shared the search key with the e-cash register before purchase, this key must be submitted.
- If no key was shared, the e-cash register must generate a unique search key for the receipt and submit it.
- Receipts that involve the customer must be encrypted and submitted to the receipt repository.

If a search key has been submitted, the encrypted receipt is forwarded to the receipt repository; otherwise, it is only transmitted to NAV.

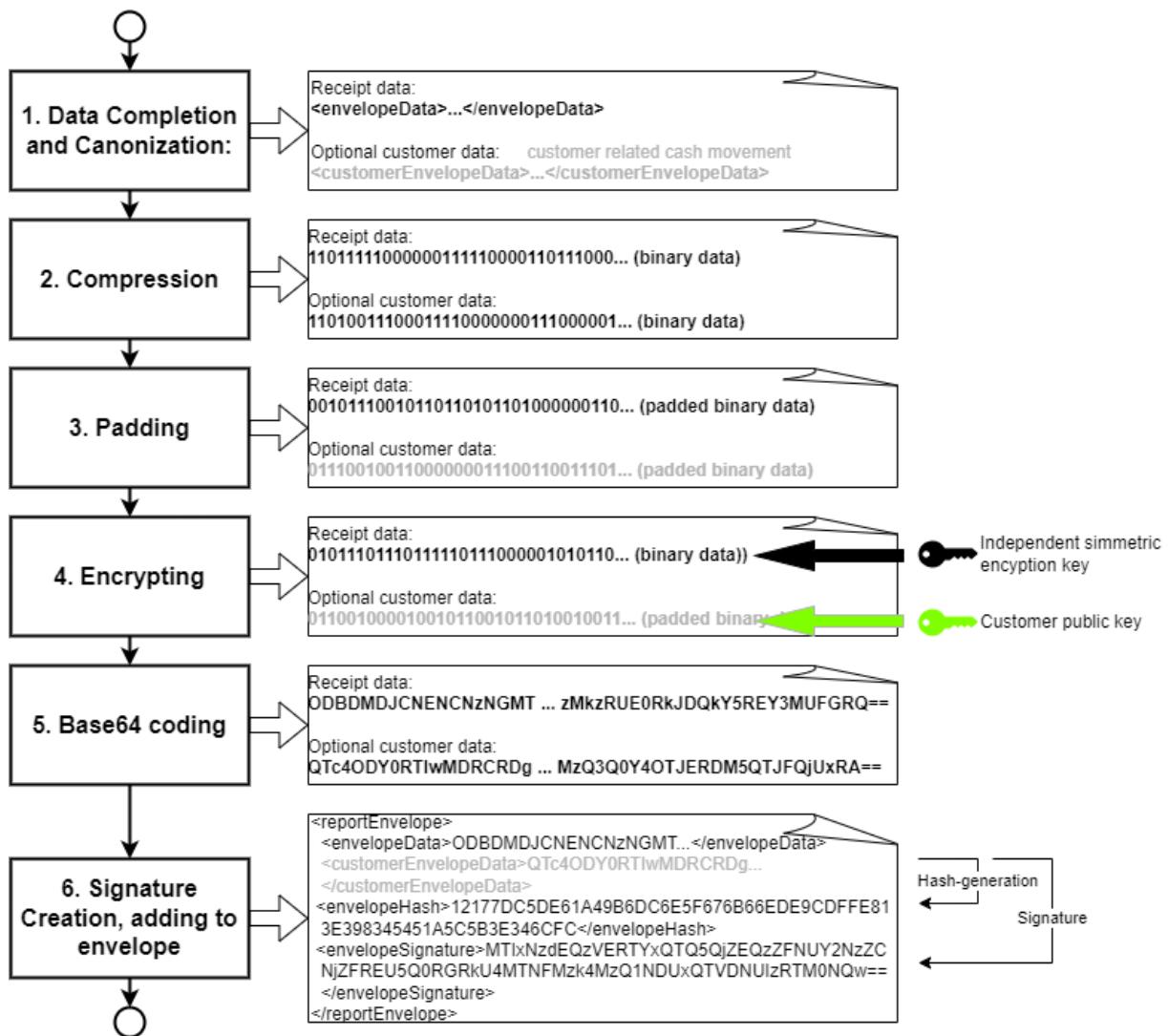
All receipts must be submitted in signed form, as described in the "[Creating Signatures](#)" chapter.

For receipts involving the customer, the e-cash register must provide visual and audio notifications of successful submission. These notifications must be displayed when NAV-I returns a successful response.

- The visual notification must be a GIF file, which may be animated.
- The audio notification must use a WAV file.
- NAV will publish these notification files on its website.

5.3.2 Compilation of the report document envelope

The envelope compilation process is shown in the following diagram:



Steps of compilation of the ReportEnvelope:

1. Data Completion and Canonization: Before sending, the required data structure must be completed according to the specifications in the "[Description of business data content \(XSD model types and elements\)](#)" section:
 - In the case of an event that does not contain customer data (i.e., not affecting the customer):
 - Receipt Data (EnvelopeData):
 - Cash Register Opening Report: CashRegisterOpenBalanceReport, or
 - Daily Sales Report: DailyCashFlowReport, or
 - Cash Register Report: CashRegisterReport, or
 - Cash Flow Report: CashFlowReport, or
 - Other Report: OtherReport
- In the case of a report that must also be provided to the customer – such as a cash flow receipt or other report:
 - Receipt Data (EnvelopeData):
 - Cash Flow Report: CashFlowReport, or



- Other Report: OtherReport
- Customer data (CustomerEnvelopeData):
 - Independent symmetric encryption key: decryptKey
 - Attachment: ReportAdditional

The completed data structure must be converted to a canonical form (<https://www.w3.org/TR/xml-c14n11/>) and validated according to the methodology issued by NAV. If the validation detects an error, the errors must be corrected before submission.

2. Compression: The data produced in the previous step must be compressed according to the method described in the "[Compression](#)" chapter. The result of compression is binary data
3. Padding: The binary data obtained from compression must be prepared for the next step using PKCS#7 padding.
4. Encryption: The previously generated data must be encrypted using a separate symmetric key for report data (EnvelopeData) and the customer's asymmetric key pair public key for optional customer data, following the encryption method described in the "[Encryption](#)" chapter. The result of the encryption operations is binary data. The envelope data to be sent to NAV (ntcaReportEnvelope) does not need to be encrypted, so this step is skipped in that case.
5. Base64 encoding: The binary data obtained from the compression step must be base64 encoded.
6. Generating signature: The base64-encoded report data from the previous step must be placed in the envelopeData field of the ReportEnvelope, while the optional customer data package must be placed in the customerEnvelopeData field. From these, envelopeHash and envelopeSignature must be generated according to the method described in the "[Creating Signatures](#)" section.

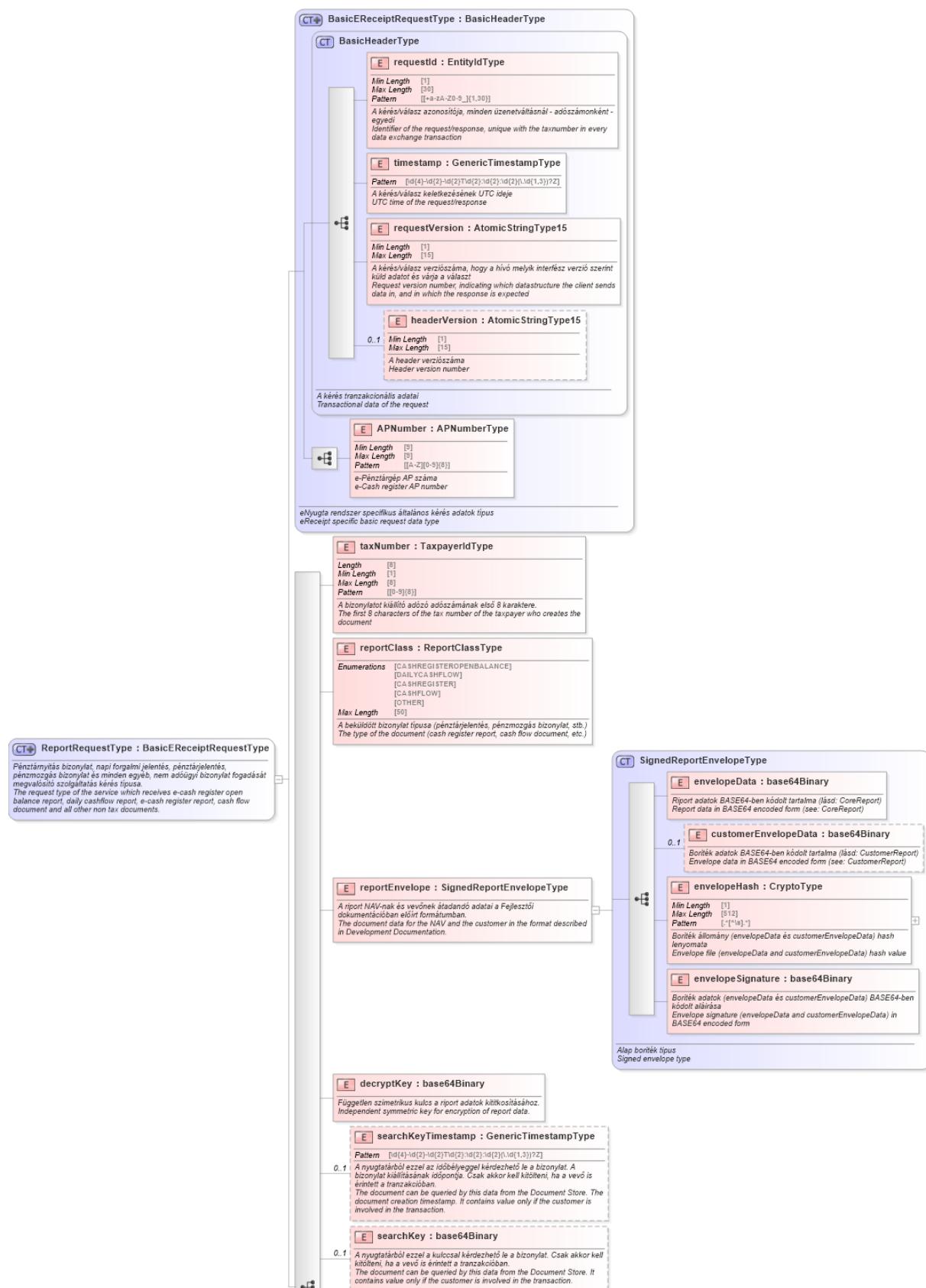
5.3.3 Technical description of the service

The receipt submission is implemented by the "report" service.

- Context root: /eReceipt/v1
- URL: /report
- Request object: ReportRequest. The technological description of the service request object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- Responseobject: ReportResponse. The technological description of the service response object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.



ReportRequest request object:

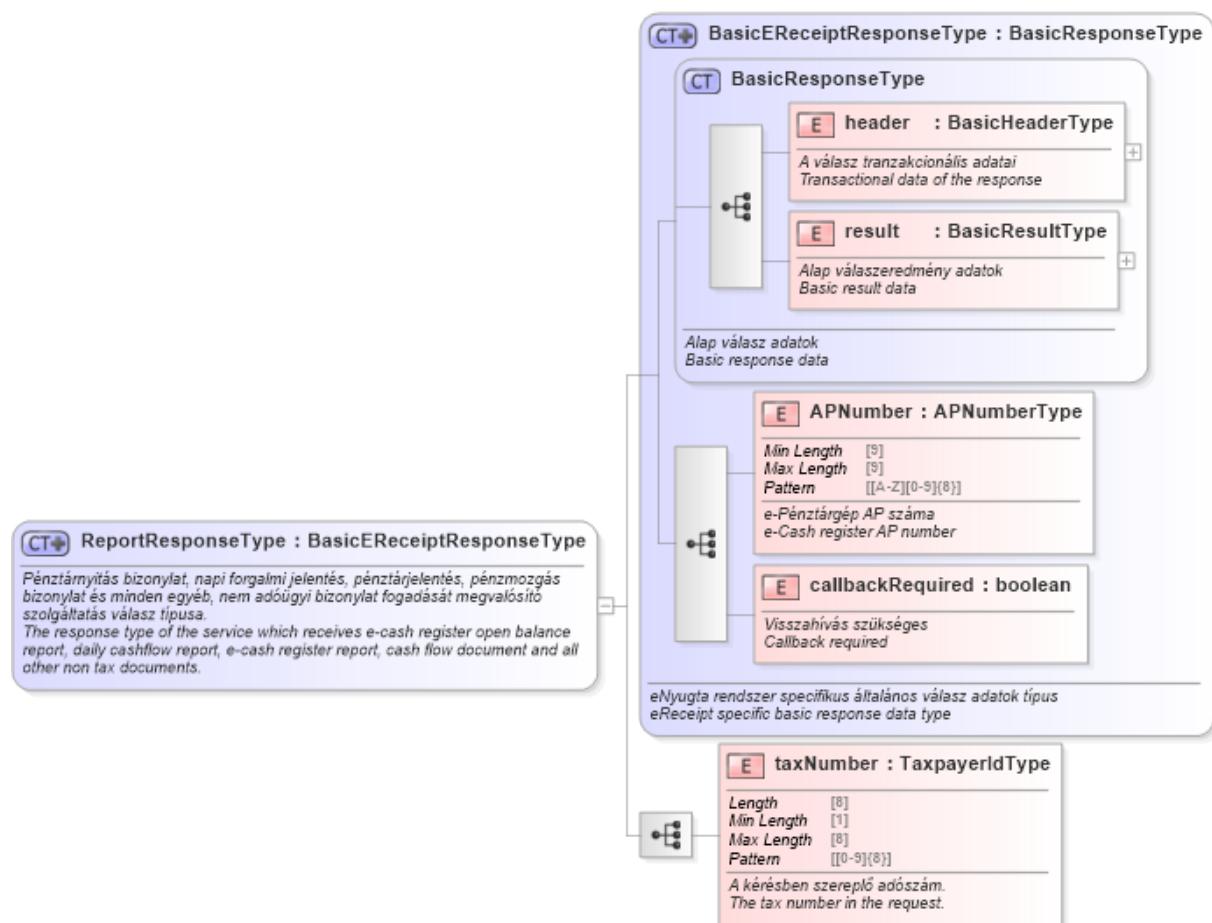




<p>E qCodeExpired : boolean</p> <p>A QR generálása a feljelzői dokumentációban meghatározott időtervben belülre történhet. Ez esetben a QR kód már lejárt.</p>
<p>E offlineCreated : boolean</p> <p>A hálózati elérhetetlensége miatt megszűnés előtt az érintett bizonyságot és adatszolgáltatásokat haladéktalanul el kell külni, a késleltetett küldés tényét ennek a mezőnek a kitöltésével kell jelezni. Whether the document was created offline. When the communication network is online again, the document has to be sent immediately with this flag.</p>
<p>E cashRegisterSignCertificate : CertificateType</p> <p>A e-pénztárgép aláíró tanúsítvány DER formátumban. A tanúsítvány privát kulcsával kerülnek aláírásra az e-pénztárgép által kiadott szemétek. Az aláíró kulccsal kapcsolatos követelmények a Feljelzői dokumentáció „Aláírás képzése” fejezetében találhatók. The sign certificate of the e-cash register in DER format. The private key of this certificate has to be used to sign the messages of the e-cash register. The requirements of the sign key can be found in the “Signing” chapter of the Development Documentation.</p>
<p>E recordCounter : nonNegativeInteger</p> <p>Az aktualisan beküldött bizonyság recordCounter értéke. Leírását lásd a Feljelzői dokumentáció „NAV ellenőrző kód képzése” fejezetében. The recordCounter value of the actual document. See further information in the “Generate NAV verification code” chapter of the Development Documentation.</p>
<p>E lastRecordCounter : nonNegativeInteger</p> <p>Az e-pénztárgépen utoljára kiadott bizonyság recordCounter értéke. Leírását lásd a Feljelzői dokumentáció „NAV ellenőrző kód képzése” fejezetében. The recordCounter value of the last document in the e-cash register. See further information in the “Generate NAV verification code” chapter of the Development Documentation.</p>
<p>E ntcaVerificationCode : SHA256Type</p> <p>Min Length [1] Max Length [64] Pattern [[0-9A-F]{64}]</p> <p>NAV ellenőrző kód. Leírását lásd a Feljelzői dokumentáció „NAV ellenőrző kód képzése” fejezetében. Ntca verification code</p>
<p>E reportEnvlopVersion : AtomicStringType15</p> <p>Min Length [1] Max Length [15]</p> <p>0..1 Amennyiben a beküldött boríték tartalmát egy régebbi interfejsz verzió szerint állította elő a pénztárgép (újaküldés esetén), mint a jelenlegi interfejsz verziót (requestVersion), akkor annak verziószámát ír meg kell. If the contents of the submitted envelope were produced by the cash register according to an older interface version (in case of resending) than the current interface version (requestVersion), then its version number must be entered here.</p>
<p>E sendMissingDocumentProcessId : ProcessIdentifierType</p> <p>Min Length [10] Max Length [10] Pattern [[A-Z0-9]{10}]</p> <p>0..1 A rendelkezésre álló sorozatúszó a SEND_MISSING_DOCUMENT felszolgáltatás végrehajtásakor történik, akkor itt kell megadni az abban kapott folyamat azonosítót. Normál beküldés esetén nem kell megadni. If the report is sent due to the execution of the SEND_MISSING_DOCUMENT command, the process ID received in it must be entered here. It is not required for normal submission.</p>



ReportResponse response object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

5.4 E-Cash register status report

This service allows the e-cash register to provide data to NAV regarding its operation, status, and communication. Through this service, operational information, system status, sales statistics, time settings, and geographical location data of the e-cash register are sent. The data must reflect the current status of the e-cash register, and for event-type data, it must include events that have occurred since the previous status report. The status report must be sent whenever a significant state change occurs, upon receiving and completing a process confirmation from the Communication Manager (e.g., taxpayer data update, VAT database update, software update), but at least once every 24 hours.

If the communication network is unavailable, the e-cash register may retry submitting the status report up to two times. If these attempts fail, the status report can be ignored.

5.4.1 Business description of the service

The service must be called in the following cases:

- When the e-cash register receives taxpayer data updates
- After executing a taxpayer data update
- When the e-cash register receives VAT database updates
- After executing a VAT database update
- When the e-cash register receives a software update



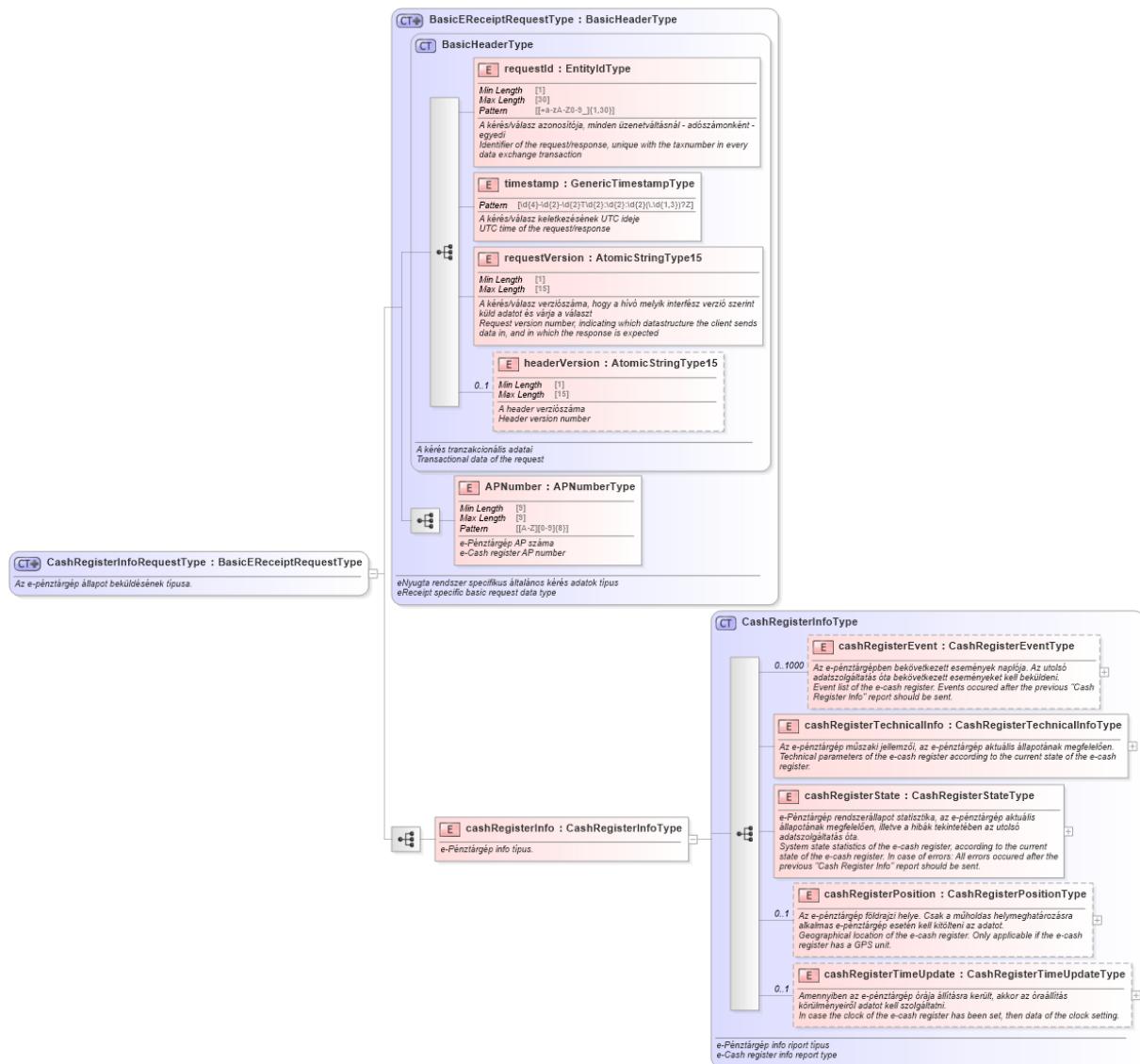
- After executing a software update
- When an e-cash register blocking instruction is executed
- When an e-cash register unblocking instruction is executed
- After opening a fiscal day
- After closing a fiscal day
- Upon request from the Communication Manager
- If 24 hours have passed since the last status report submission
- When the cash register is turned on (logging a "PIN" event)
- The geographical location of the e-cash register capable of positioning differs by at least 20 (twenty) arcseconds from its previously recorded location.

5.4.2 Technical description of the service

The status report submission is implemented through the "cashRegisterInfo" service.

- Context root: /eReceiptMgmt/v1
- URL: /cashRegisterInfo
- Request object: CashRegisterInfoRequest (technical details are provided in the "[Description of business data content \(XSD model types and elements\)](#)" section).
- Response object: CashRegisterInfoRequest (technical details are provided in the "[Description of business data content \(XSD model types and elements\)](#)" section).

CashRegisterInfoRequest request object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

The `CashRegisterInfoResponse` response object does not contain any endpoint-specific data other than header and result data

5.5 Communication manager

The purpose of this service is to regulate communication between the e-cash register and the NAV-I central system, publishing NAV-initiated communications towards e-cash registers.

5.5.1 Business description of the service

For previous online cash registers, the tax authority required the registers to regularly establish a connection with the NAV central system at predetermined intervals. During these communication events, when the cash register contacted the NAV system, it received instructions from NAV regarding the necessary communication actions.



For e-cash registers, the regulations have changed: every single document generated by the e-cash register must be immediately forwarded to the NAV-I central system. Each receipt or report must be sent to the central system as soon as it is created.

When the NAV central system receives a receipt or report from the e-cash register, it responds by indicating whether additional services need to be invoked within the NAV system. If additional service calls are required, the e-cash register must invoke this Communication Manager service. This service returns a list of URLs specifying which services the e-cash register must call, in a specific sequence. Since some of these services may depend on others, the e-cash register must invoke them sequentially, in the specified order.

In case of a failed service call, the service can either be retried by the e-cash register or be re-invoked by NAV-I at a later time.

If the e-cash register does not send any receipt or report for more than 30 minutes, it must call this service after 30 minutes have passed.

For hardware-based e-cash registers, there must also be a function allowing manual invocation of this service by the operator. This service call can be used to test communication between the e-cash register and the NAV-I system. Cloud-based e-cash registers do not require this manual invocation function.

5.5.2 Technical description of the service

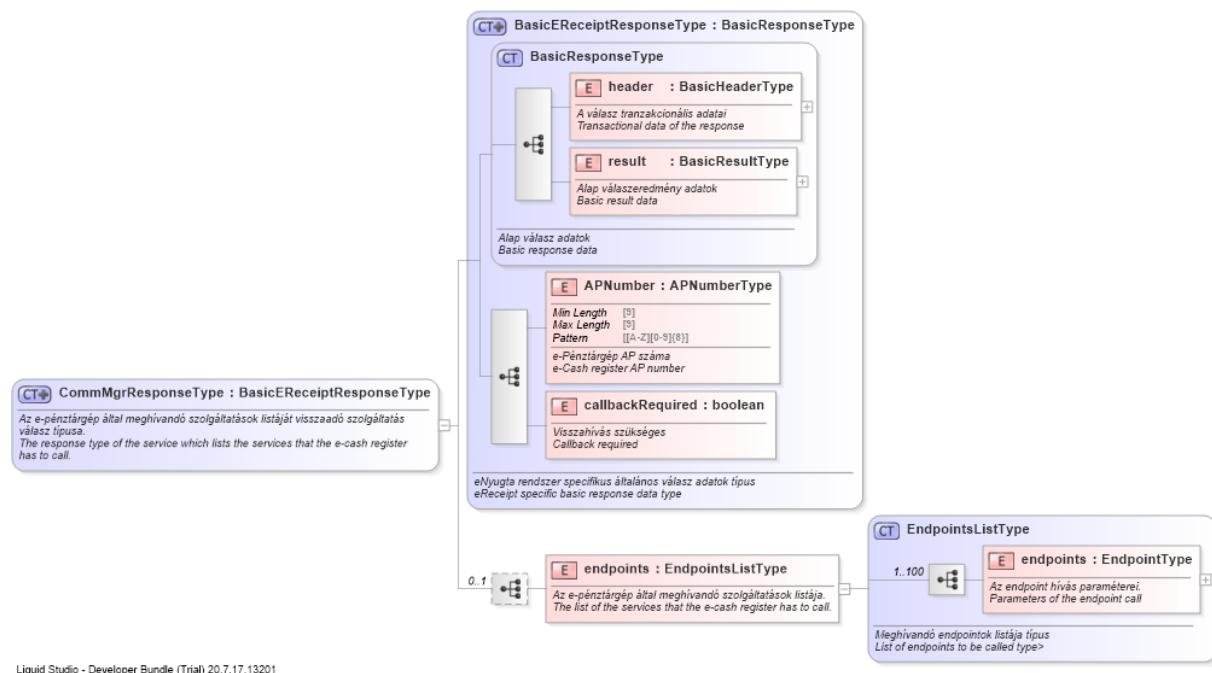
The service is implemented as the "CommMgr" service.

- Context root: /eReceiptMgmt/v1
- URL: /commMgr
- The request object is CommMgrRequest, and its technical details can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- The response object is CommMgrResponse, and its technical details can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- .

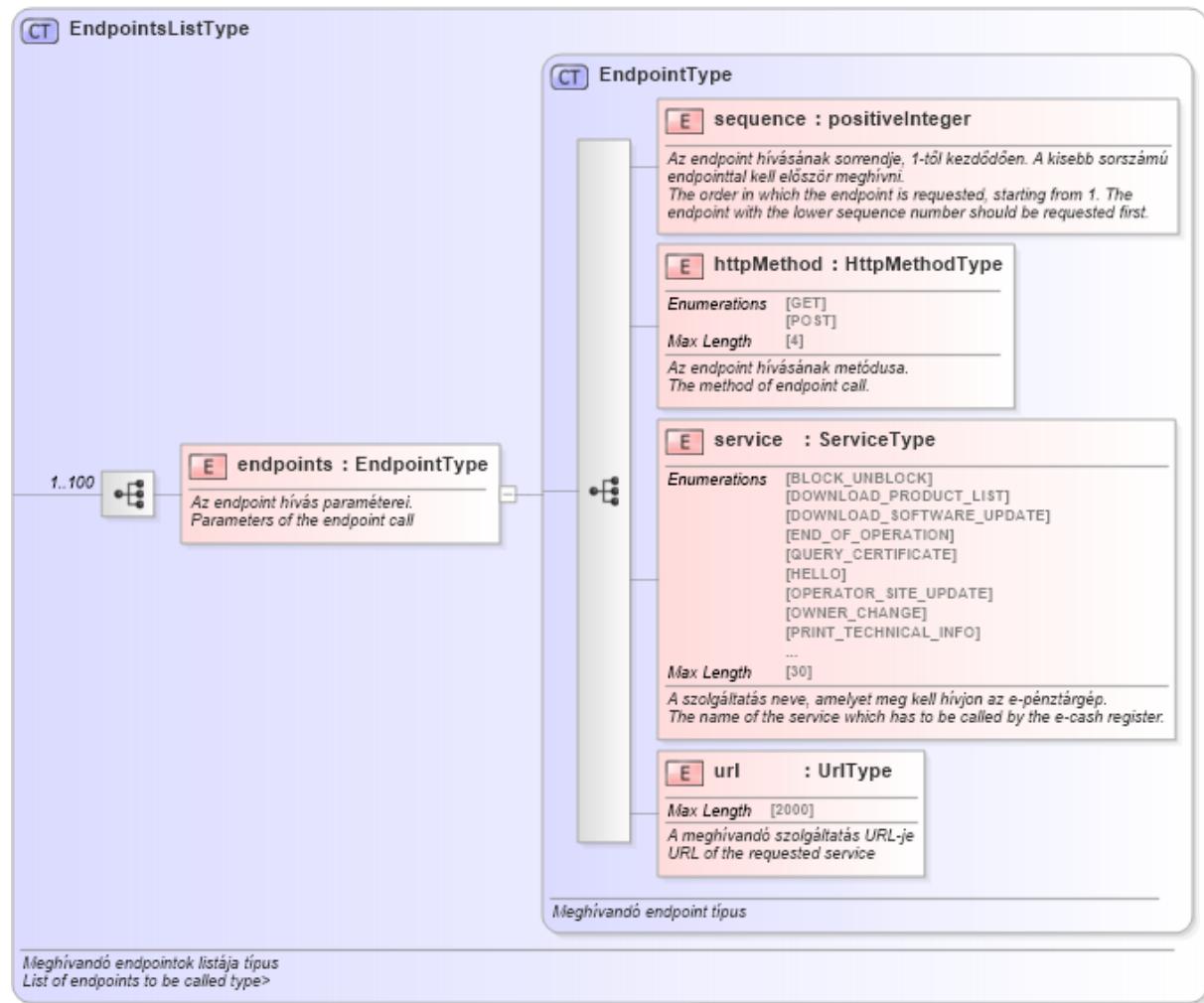
The CommMgrRequest call object only contains header data, as described in the "[General Technical Data](#)" section.



CommMgrResponse response object:



If the e-cash register needs to call at least one service, the endpoint list in the response object will not be empty, and its elements will consist of the following objects:



5.6 Taxpayer data query

The service is used to update taxpayer data registered with NAV, including operator or operational site data stored on the e-cash register.

5.6.1 Business description of the service

Through the interface, the e-cash register receives information about when (year, month, day, hour, minute, second) to apply updated operator and operational site data. In the case of dual-business e-cash registers, this includes both the fuel owner and the store operator.

The call of the service must be initiated by the e-cash register either before opening a new fiscal day or if NAV's central system has indicated the need for an update. NAV signals this requirement through the Communication Manager service, the details of which are described in the "[Communication Manager](#)" section.



NAV Data received from NAV's central system will be in the following format:

- For operator data: address details, full taxpayer name, short taxpayer name, taxpayer ID number, and VAT group membership identifier (if applicable).
- For dual-business fuel station e-cash registers: address details, full taxpayer name, short taxpayer name, taxpayer ID number, and VAT group membership identifier (if applicable) for both the fuel owner and the store operator.
- For operational site data: address details, full store name, and short store name.

The received message always contains both operator and operational site data, but data for a second operator is only included in the case of dual-business e-cash registers.

When generating electronic receipts, the e-cash register may insert line breaks or hyphen characters in the header data to ensure proper formatting. However, the data displayed in the receipt header must match the information received in the message, aside from these formatting adjustments.

The e-cash register must store the received data until execution. The stored data must be included in status reports under "pending taxpayer data validation" to confirm to NAV-I that the e-cash register has received and acknowledged the update request. After storing the data, the e-cash register must immediately send a status report.

The taxpayer data update must be executed while the e-cash register is in a closed fiscal day. If the update is not validated, the e-cash register must not open a new fiscal day beyond the specified timestamp in the received message.

5.6.2 Technical description of the service

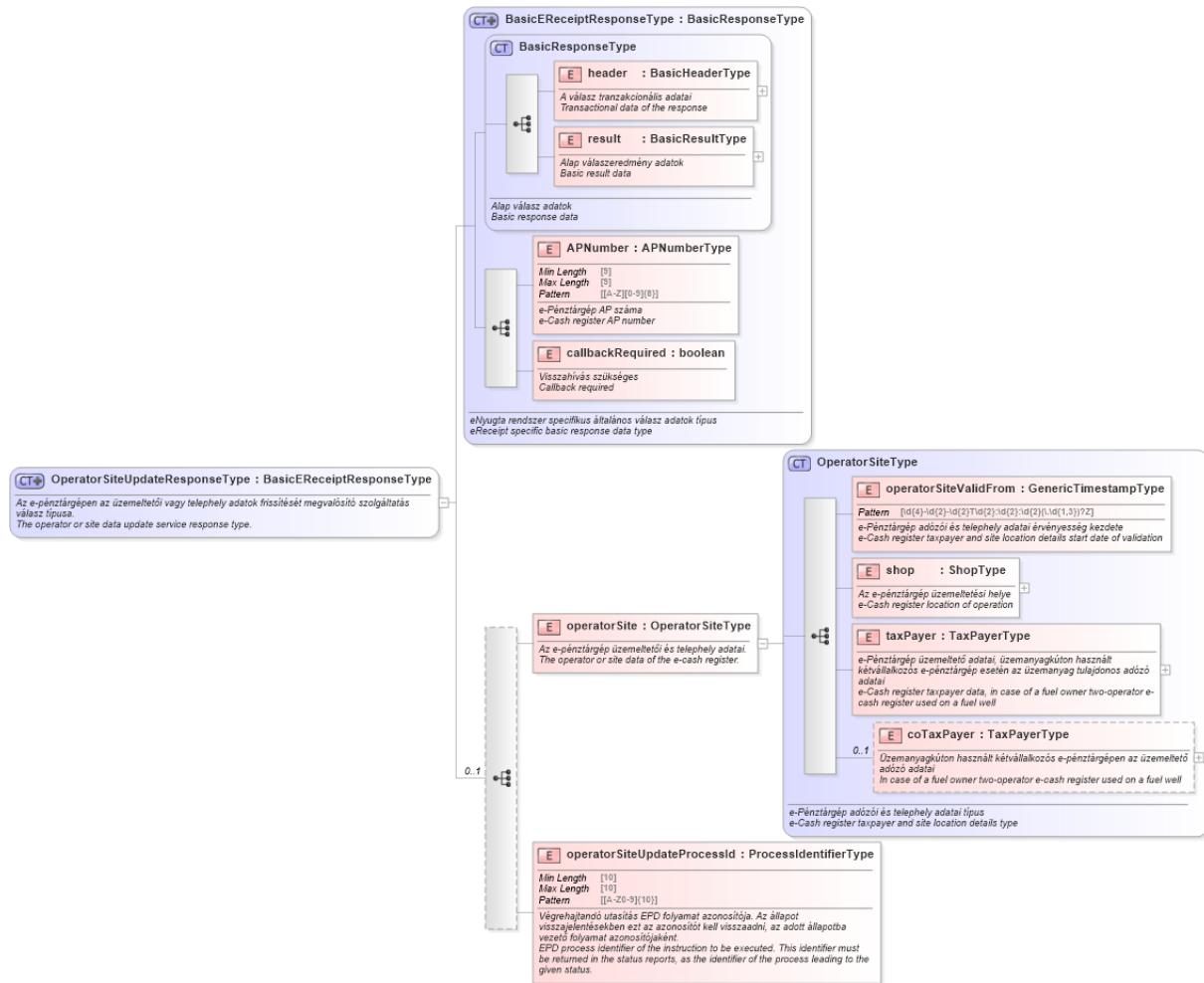
The taxpayer data query is implemented by the "operatorSiteUpdate" service.

- Context root: /eReceiptMgmt/v1
- URL: /operatorSiteUpdate
- Request object: OperatorSiteUpdateRequest. The technical description of the service request object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- Response object: OperatorSiteUpdateRequest. The technical description of the service response object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.

The OperatorSiteUpdateRequest call object contains only the header data described in the "[General Technical Data](#)" section.



OperatorSiteUpdateResponse response objects:



5.7 VAT master data query

This service is used for managing and modifying the use of specific sales registers and their associated tax rates and tax contents.

5.7.1 Business description of the service

Through the interface, the e-cash register receives information on the sales registers specified in the request message and the applicable tax rates and tax contents, including the exact date and time (year, month, day, hour, minute, second) from which they must be applied.

The service call must be initiated by the e-cash register, either before opening the fiscal day or when the NAV central system notifies the e-cash register. NAV can indicate the necessity of an update via the Communication Manager service, described in the "[Communication Manager](#)" section.

The data received from the NAV central system is structured as follows:

- The effective date from which the tax rates and tax contents must be applied.



-
- Sales register codes (labels) used to ensure the separate registration of sales and other transactions.
 - VAT rate is expressed as a percentage, without the percentage symbol, using a decimal point if necessary.
 - VAT content corresponding to the VAT rate, expressed as a percentage, without the percentage symbol, using a decimal point if necessary.
 - The VAT rate value expressed as an integer, including the percentage symbol.

The message received always contains the full set of sales registers and VAT rates, regardless of whether their data has changed. The modification of sales registers and labels is carried out based on the NAV message.

The e-cash register must store the received data until execution. The stored data and process identifier must always be included in the status reports, thereby confirming to NAV-I that the e-cash register has received the instruction and the data. After storing the data, the e-cash register must immediately send a status report.

The VAT registry update must be executed by the e-cash register on a closed fiscal day, ensuring that the device cannot start a new fiscal day after the specified date without validating the update.

5.7.2 Technical description of the service

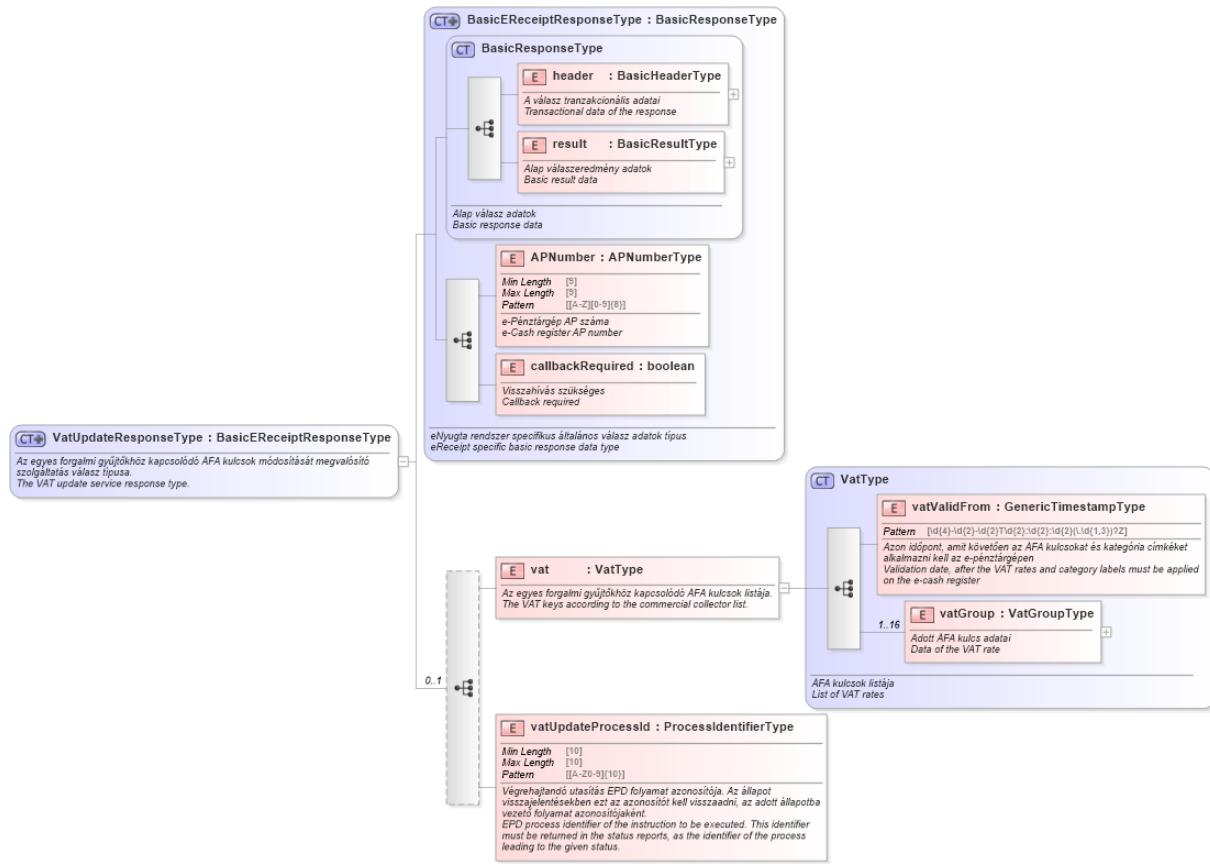
The VAT registry query is implemented through the "vatUpdate" service.

- Context root: /eReceiptMgmt/v1
- URL: /vatUpdate
- Request Object: VatUpdateRequest. The technical description of this service's request object is found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- Response Object: VatUpdateResponse. The technical description of this service's response object is found in the "[Description of business data content \(XSD model types and elements\)](#)" section.

The VatUpdateRequest call object only contains the header data described in the "[General Technical Data](#)" section.

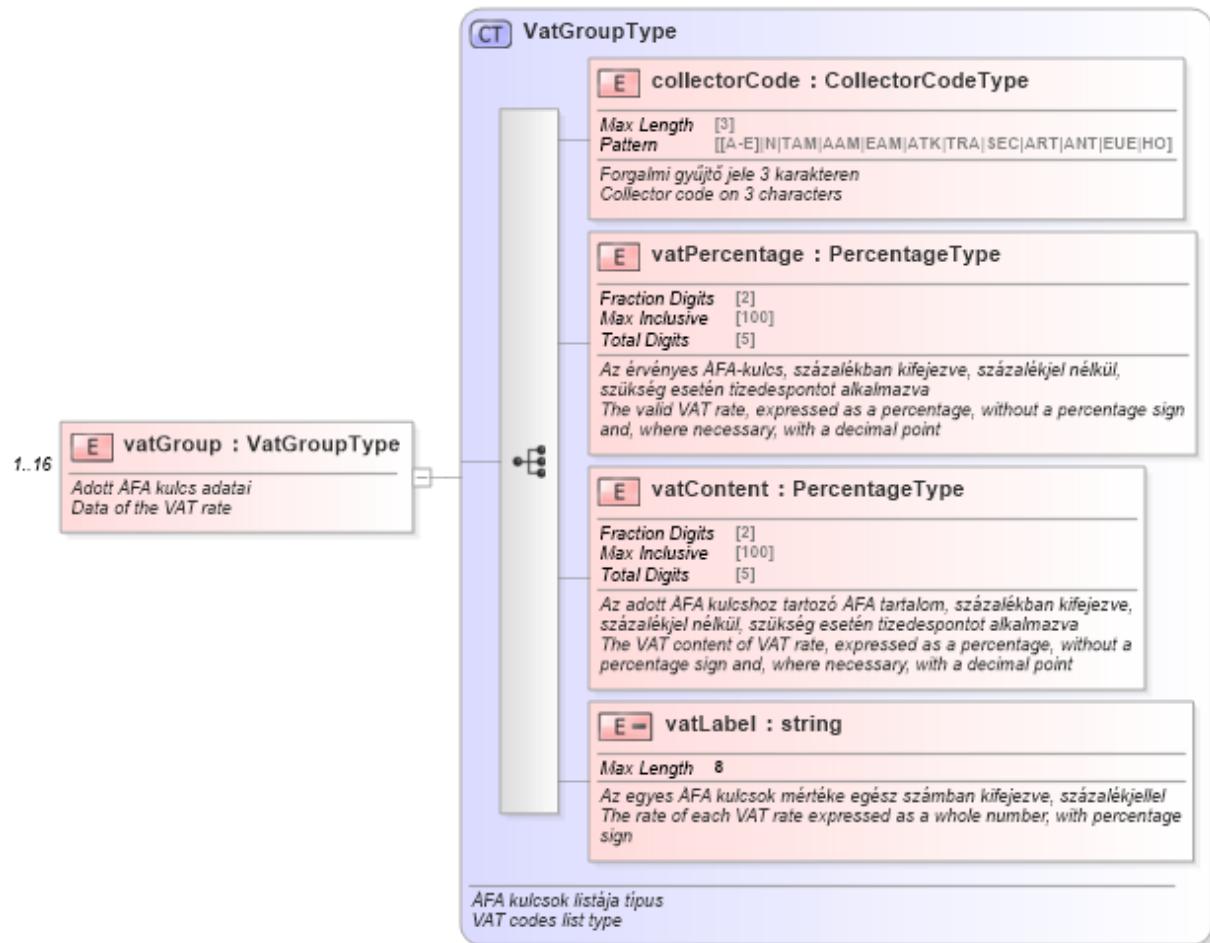


VatUpdateResponse response object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

The VAT rates returned in the response are listed in objects with the following structure:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

5.8 E-cash register blocking/unblocking

The service is used to place the e-cash register in a blocked state or to lift the blocking of the e-cash register.

5.8.1 Business description of the service

According to regulations applicable to e-cash registers, the NAV can block e-cash registers for various reasons, and once the reasons cease to exist, the blocking can be lifted. The service request must be initiated by the e-cash register either before opening a new fiscal day or when the NAV central system signals the e-cash register about the necessity of blocking. The NAV may signal the necessity of blocking through the Communication Manager service, described in the "[Communication Manager](#)" section. The blocking can also be initiated by the e-cash register operator as part of the termination process of operations, as described in the "[Termination of operations](#)" section. The e-cash register may request the lifting of the blocking as part of the "[Resumption of operation](#)" section.

The reason for blocking may be one or more of the following:

- The e-cash register is in a terminated state.
- The e-cash register initiated its own blocking (e.g., due to technical issues).
- Blocking initiated by a tax auditor.
- Technical blocking initiated by NAV (e.g., due to legal status changes).



-
- Blocking initiated by the mobile service provider (e.g., due to unpaid invoices).

The data received from the NAV central system can have two possible values: BLOCK (blocking) or UNBLOCK (lifting of blocking).

The received command must be executed immediately as instructed in the message. If an open receipt exists, the execution should take place only after the receipt is closed.

If the blocking or unblocking is initiated by the e-cash register, the operation can only be performed if the NAV response contains the BLOCK (blocking) or UNBLOCK (lifting of blocking) command. Otherwise, the blocking status of the e-cash register cannot be changed.

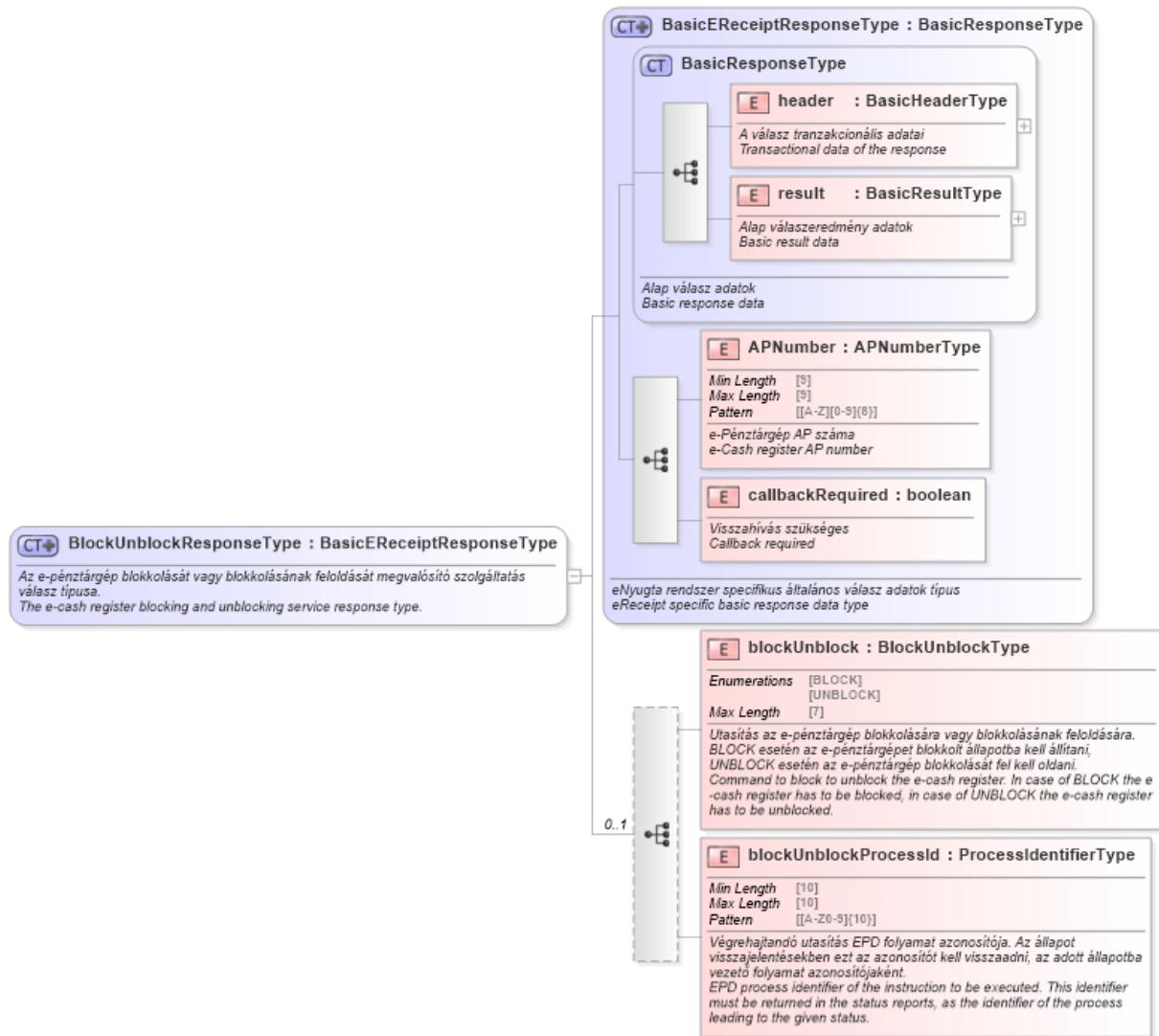
5.8.2 Technical description of the service

The e-cash register blocking/unblocking is implemented by the blockUnblock service.

- Context root: /eReceiptMgmt/v1
- URL: /blockUnblock
- **Request object:** BlockUnblockRequest. The technical description of the service request object is detailed in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- **Response object:** BlockUnblockResponse. The technical description of the service response object is detailed in the "[Description of business data content \(XSD model types and elements\)](#)" section.

The BlockUnblockRequest call object contains only the header data described in the "[General Technical Data](#)" section.

BlockUnblockResponse response object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

5.9 Sending technical information

Through the Communication Manager service, the NAV can instruct the e-cash register to call the technical information sending service.

5.9.1 Business description of the service

By calling the service, the NAV central system can instruct the e-cash register to immediately display the message returned by the service on its screen and optionally print it on its printer after the first daily closing following the receipt of the message. The message received from the NAV central system can be up to 4096 characters long and may only contain letters, numbers, spaces, question marks, exclamation marks, periods, commas, hyphens, parentheses, square brackets, underscores (_), asterisks (*), percentage signs (%), equal signs (=), plus signs (+), section signs (§), slashes (/), backslashes (), and at signs (@).



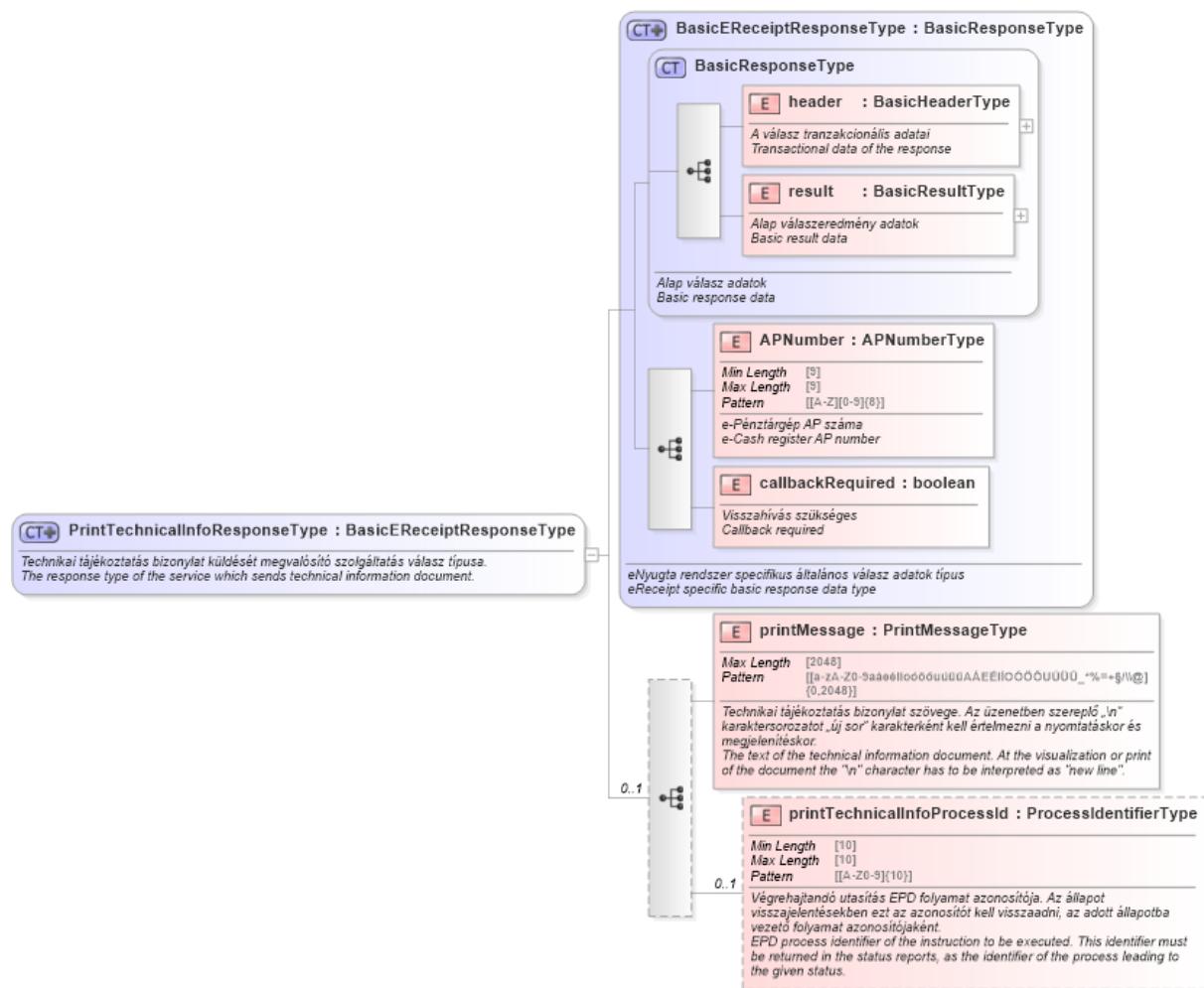
5.9.2 Technical description of the service

The sending of technical information is implemented by the "printTechnicalInfo" service.

- Context root: /eReceiptMgmt/v1
- URL: /printTechnicalInfo
- Request object: The technological description of the service request object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.
- Response object: The technological description of the service response object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.

The PrintTechnicalInfoRequest call object only contains the header data described in the "[General Technical Data](#)" section.

A PrintTechnicalInfoResponse response object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201



5.10 Software update

The e-cash registers must be prepared for software updates to be installed on them. The front-end software update of a cloud-based e-cash register must be performed from the application store independently of the NAV. The software update of the cloud-based (optional) back-end service takes place during the maintenance window defined by the operator's internal rules after the successful authorization of the version to be installed. Each software version must be registered with NAV for both the front-end and the back-end. After a software update, the software must send a status report (CashRegisterInfo) for validation. The NAV-I verifies the report data. If the submitted data does not comply, NAV blocks the software using the e-cash register blocking service.

5.10.1 Business description of the service

The e-cash register must be capable of remotely updating its software components or the entire software at the initiative of NAV-I or upon the operator's instruction. The e-cash register can only perform software updates on a closed fiscal day.

5.10.1.1 Hardware-based e-cash register

A software update for a hardware-based e-cash register refers to changes affecting the AE software. The availability of a new software version for hardware-based e-cash registers is indicated by the NAV-I central system to the e-cash register. If new software is available for download, the central system initiates its download via the Communication Manager service. Through the Communication Manager service call, the e-cash register receives information on which link to use to download the software update, the deadline for applying the update, as well as the hash required to verify the downloaded file and the hash calculation algorithm. The URL received in the response must be called using the HTTPS protocol with the GET method. After downloading, the hash must be verified to ensure the software update file has not been corrupted during the download. Only software updates with successful hash verification should be installed. If the hash verification fails, two additional attempts must be made to download the new software. The software installation process for a hardware-based e-cash register can only start if the e-cash register is operating from a network power source.

5.10.1.2 Cloud-based e-cash register

In the case of a cloud-based e-cash register, the front-end application update refers to downloading and installing a new software version from the application store. For the centrally running (optional) back-end, it refers to installing the authorized software version from an approved installation package (e.g., Docker image repository). In cloud-based e-cash registers, the front-end application notifies the user about the availability of an update.

5.10.2 Technical description of the service

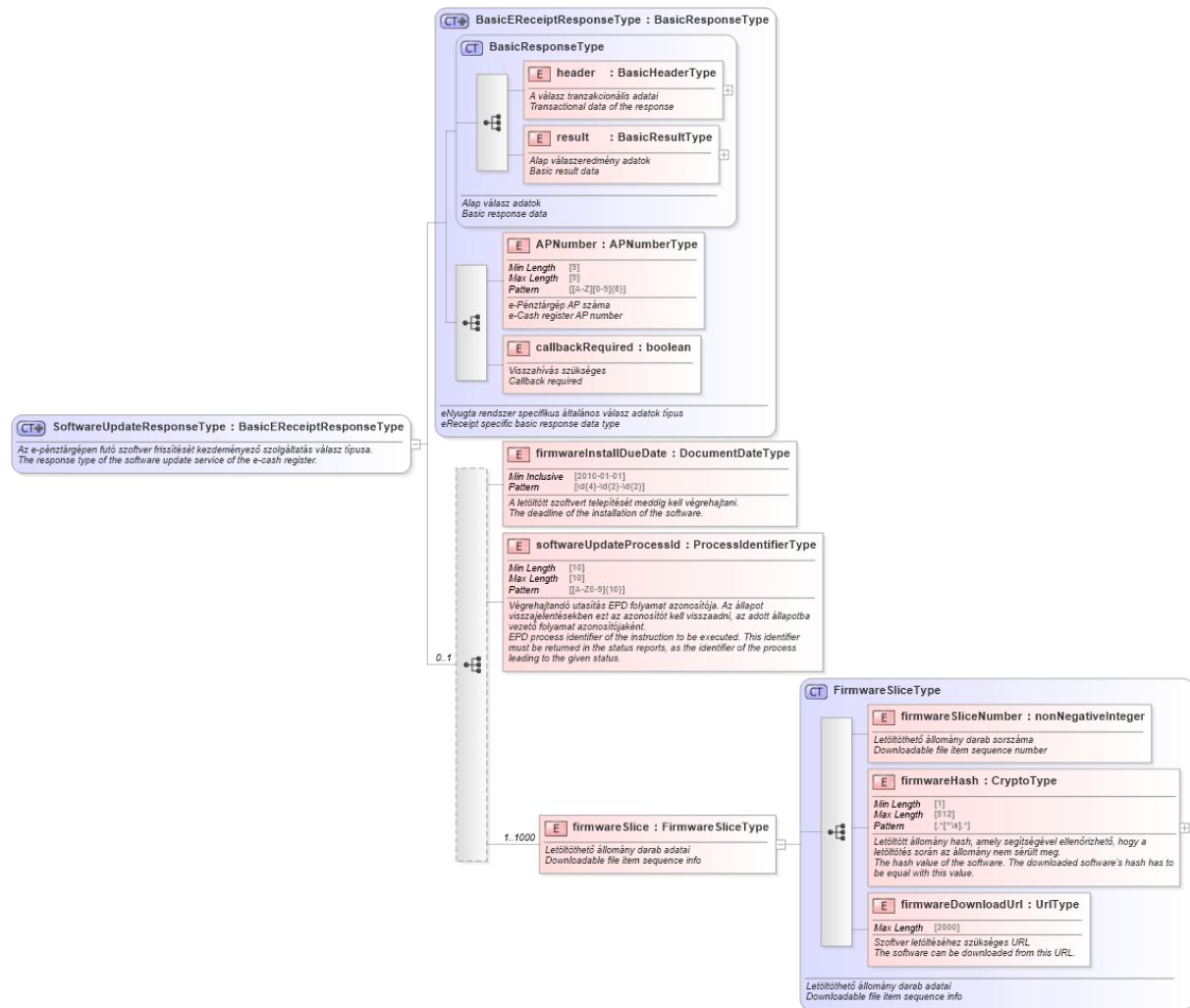
The software version update of the e-cash register is implemented by the "softwareUpdate" service.

- Context root: /eReceiptMgmt/v1
- URL: /softwareUpdate
- Request object: SoftwareUpdateRequest. The technological description of the service request object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.

- Response object: SoftwareUpdateResponse. The technological description of the service response object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.

The SoftwareUpdateRequest call object only contains the header data described in the "[General Technical Data](#)" section.

SoftwareUpdateResponse response object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

5.11 Domestic tax number verification

Domestic tax number verification service.

5.11.1 Business description of the service

The service can provide information on the validity and authenticity of a given tax number based on the NAV database, in case of a valid tax number, return the taxpayer's data, which can be used for example to provide billing address data.

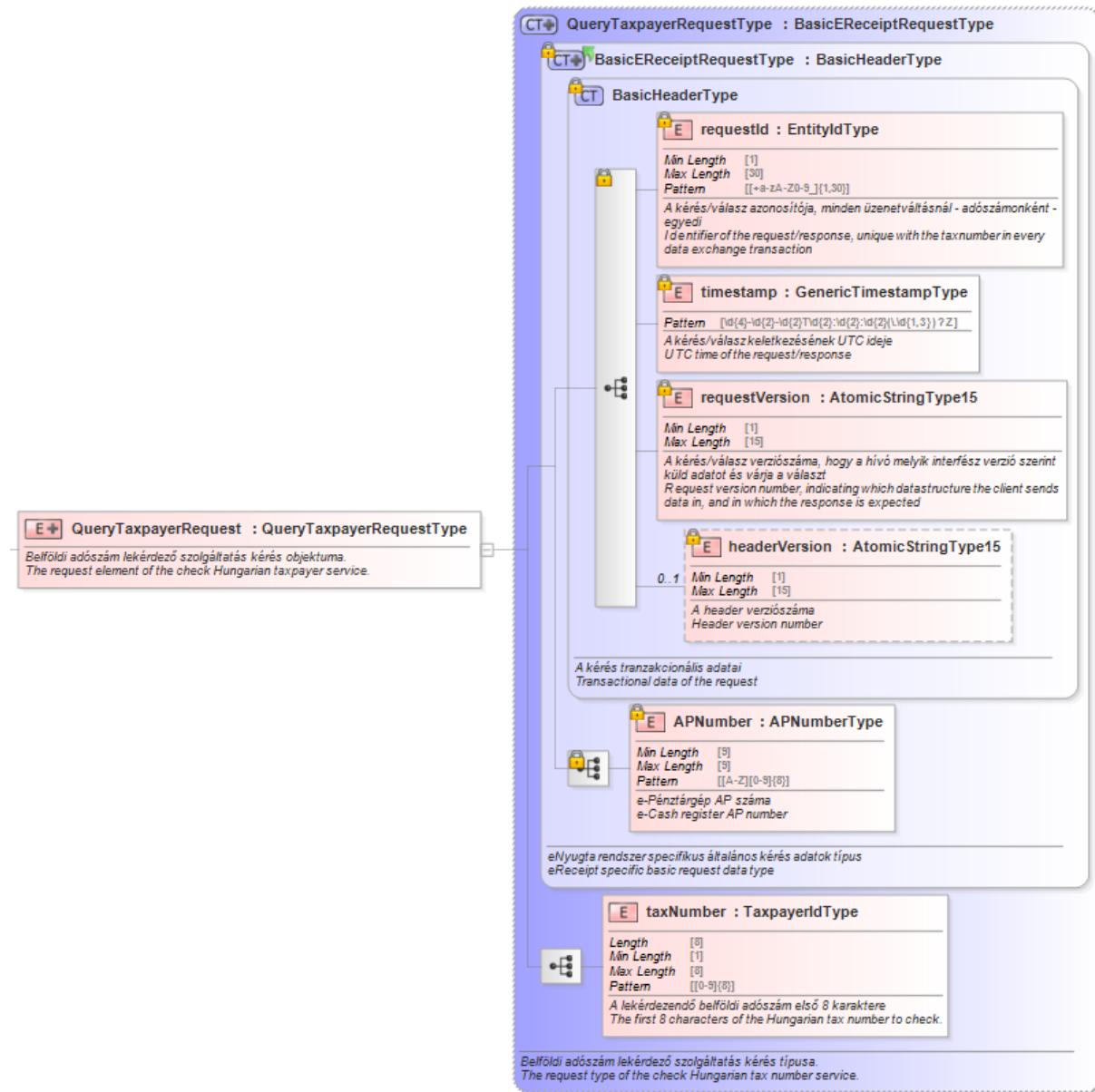
5.11.2 Technical description of the service

The domestic tax number verification is implemented by the "queryTaxpayer" service.

- Context root: /eReceiptMgmt/v1

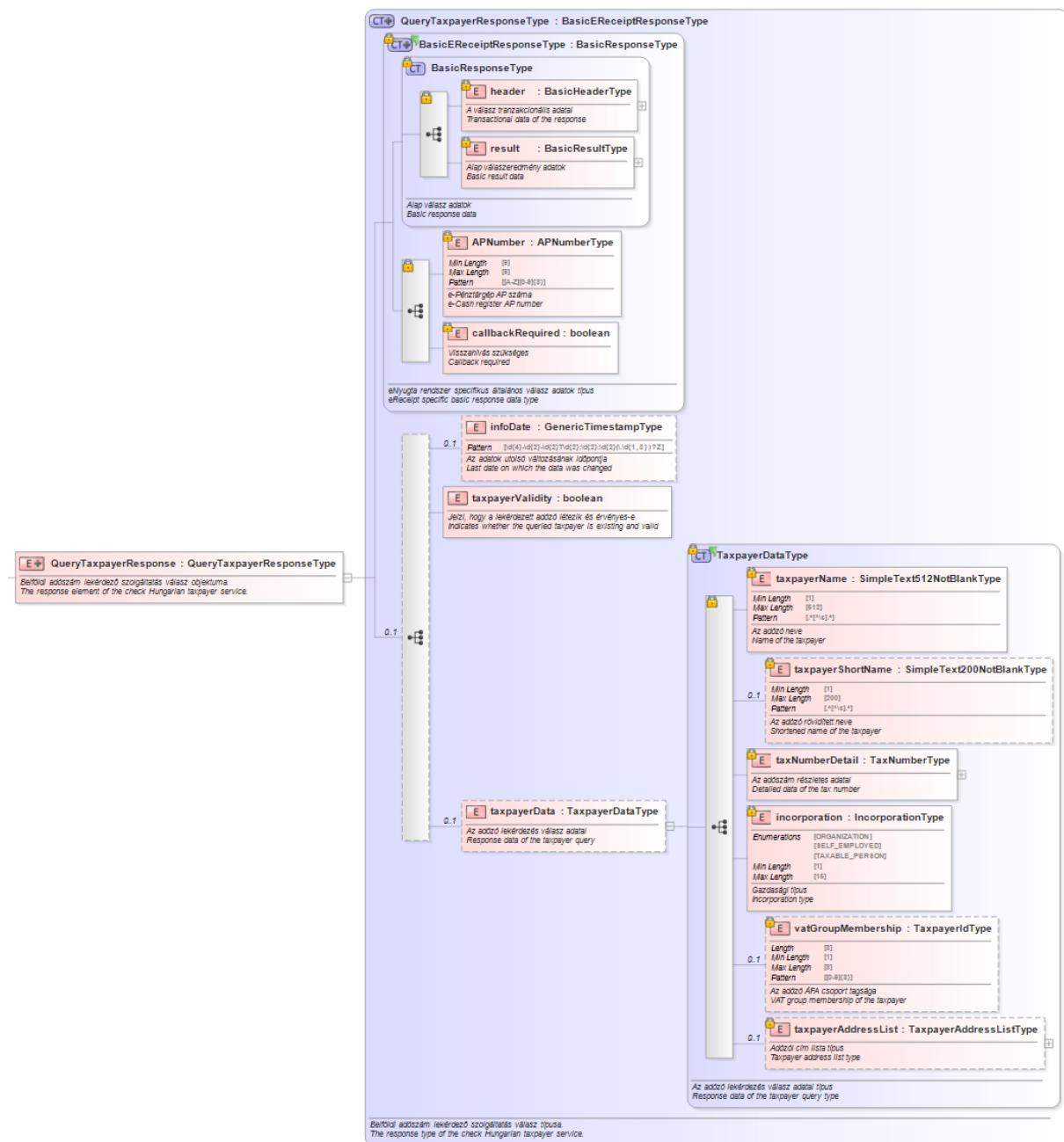
- URL: /queryTaxpayer
- **Request object:** QueryTaxpayerRequest. The technological description of the service request object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.
- **Response object:** QueryTaxpayerResponse. The technological description of the service response object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.

QueryTaxpayerRequest request object:





A QueryTaxpayerResponse response object:



5.12 Termination of operation

The termination of operation service is used to terminate the operation of an e-cash register that is in an operational state. Once the operation has been terminated, a new fiscal day cannot be opened on the e-cash register. The e-cash register can exit this state either by resuming operation (see "[Resumption of operation](#)" section) or by re-personalization (see "[Re-Personalization](#)").

5.12.1 Business description of the service

The termination of operation for an e-cash register can only be initiated by its operator through the e-cash register portal. Following this initiation, the e-cash register receives a



request to execute the process when calling the Communication Manager service. The process can only be started while the e-cash register is in an operational state.

Before initiating the termination of operation, the e-cash register must be in a blocked state. If the e-cash register is not in a blocked state, it must initiate its blocking before starting the process, as described in the "[E-Cash Register Blocking/Unblocking](#)" section.

If the e-cash register was blocked not by its own initiation (e.g., blocked by NAV due to suspension of data communication services), the termination of operation process cannot be executed.

The process can only be initiated if all previously issued fiscal data is stored on an external data carrier in possession of the operator, and all receipts and reports issued on the e-cash register have already been submitted to NAV. During the termination of operation request, the e-cash register must submit the serial number of the last issued receipt or report.

If the termination of operation process is successfully completed, the e-cash register will not be able to open a new fiscal day until either a re-personalization or the resumption of operation process is completed.

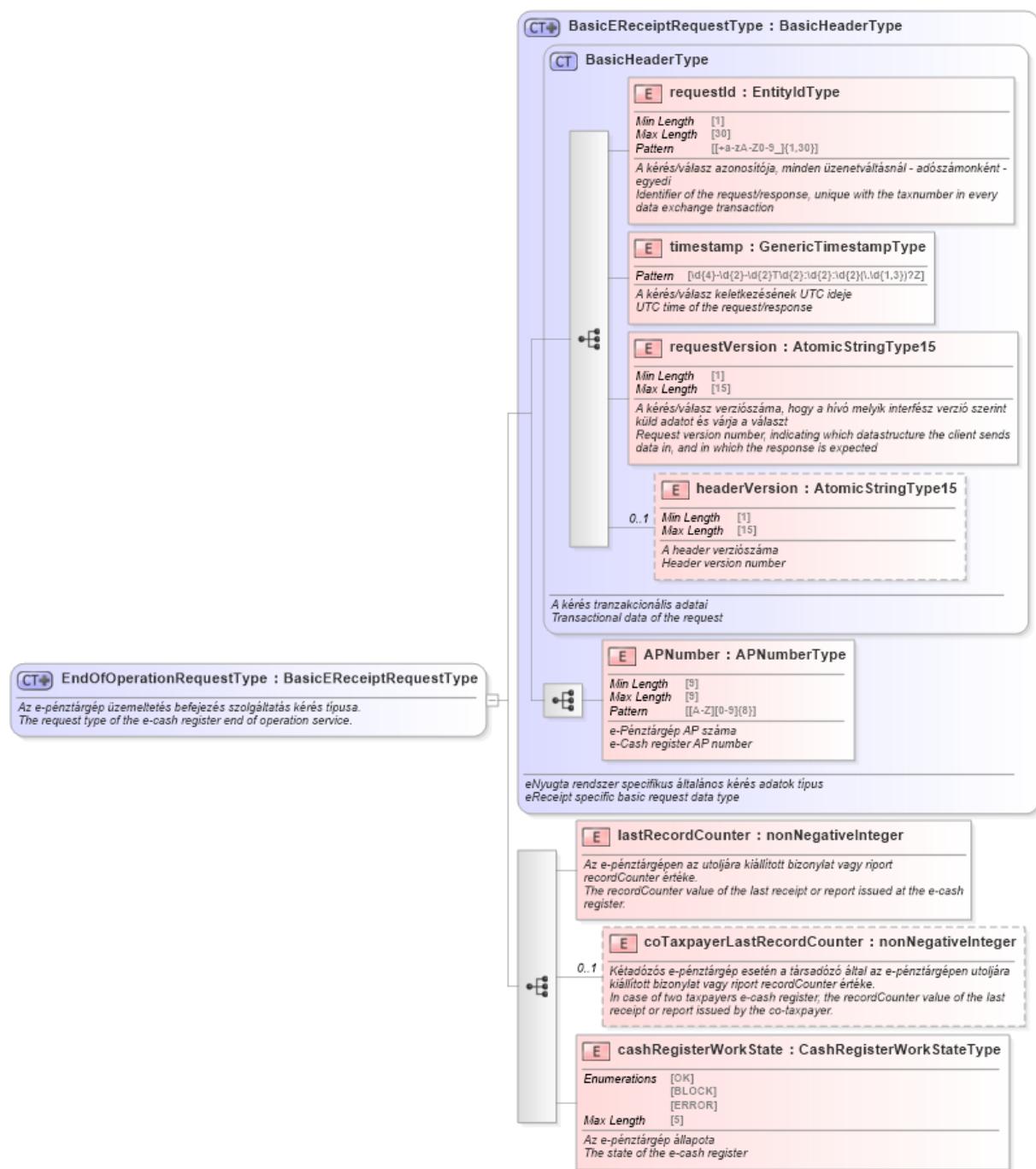
5.12.2 Technical description of the service

The termination of operation service is implemented by the "**endOfOperation**" service.

- Context root: /eReceiptMgmt/v1
- URL: /endOfOperation
- **Request object:** EndOfOperationRequest
The technical description of the request object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- Response object: EndOfOperationResponse The technical description of the response object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.



The EndOfOperationRequest request object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

The EndOfOperationResponse response object does not contain endpoint-specific data beyond the header and result data.



5.13 Resumption of operation

A service that enables the resumption of operation of the e-cash register.

5.13.1 Business description of the service

After terminating its operation, the e-cash register can resume operation by invoking this process. In this case, the existing operator does not change.

The e-cash register can resume operation by its operator initiating the unblocking process through the e-cash register portal, as described in the "[E-Cash Register Blocking/Unblocking](#)" section. No additional service needs to be called for resumption of operation. If NAV's response message includes the unblock command, the operation can continue. If NAV's response does not contain the unblock command, the e-cash register cannot resume operation.

5.13.2 Technical description of the service

To resume operation, the e-cash register must initiate the unblocking process as described in the "[E-Cash Register Blocking/Unblocking](#)" section.

5.14 Re-personalization

The service that enables the re-personalization of an e-cash register, during which the operational e-cash register is reassigned to a different operator.

Re-personalization cannot be performed on cloud-based e-cash registers.

On dual-operator e-cash registers, re-personalization is not possible. Only the fuel station operator (co-taxpayer) can be changed at the request of the fuel owner via a taxpayer data update request.

5.14.1 Business description of the service

Re-personalization can only be performed on hardware-based e-cash registers. The process can only be initiated if the e-cash register is powered by a network power source and the process "[Termination of operations](#)" has been successfully completed, all previously issued fiscal data is stored on an external data carrier in possession of the operator ad all receipts and reports issued on the e-cash register have been submitted to NAV.

During the re-personalization request, the e-cash register must submit the serial number of the last issued receipt or report. If the e-cash register has not sent all receipts and reports it has issued, the re-personalization process cannot be executed.

The e-cash register must request a new authentication and signing certificate because certificates are linked not only to the e-cash register but also to the taxpayer. The request must include a CSR file containing the request for both certificate types.

Re-personalization cannot be performed for the current operator. In this case, the operation can be continued using the "[Resumption of operation](#)" service.

Re-personalization is a multi-step process where the e-cash register must invoke several services. Calling the re-personalization service is only the first step of the process. A successful response to this request alone does not mean that the re-personalization has been successfully completed.



The steps of the re-personalization process are as follows:

- **Calling the Re-Personalization Service:** This initiates the re-personalization process.
- In the response, NAV provides the e-cash register with the new operator and operational site data, URLs for downloading the required certificates and instructions to unblock the e-cash register.
- After receiving the response, the e-cash register must update the operator and operational site data, unblock the e-cash register and download and store the two new certificates.
- Before the first download attempt, the e-cash register must wait 5 seconds. If the certificate is not available at the given URL, retry attempts must have at least a 10-second interval. If the certificates are not successfully downloaded within 5 minutes, the re-personalization process is considered unsuccessful, and the service must be called again. If reattempting, a new CSR with newly generated keys must be submitted
- The unblock command can only be executed after successfully downloading the certificates.
- Once completed, the e-cash register must call the CashRegisterInfo Report Submission Service.
- **Submitting the CashRegisterInfo Report (Calling the Report Reception Service):** The e-cash register must send operational data that already reflects the updated operator and operational site information. It must confirm that the e-cash register has been successfully unblocked.
- **Calling the Hello Service with the New Authentication Certificate:** A successful response from NAV confirms that the re-personalization was successfully executed. After receiving the successful response, the e-cash register must:
 - Delete all receipts and reports issued before the re-personalization.
 - Remove the old certificates
 - The deletion must be performed up to the recordCounter value specified at the beginning of the re-personalization process.

Re-personalization is **only considered successfully completed** when:

- All required modifications instructed by NAV have been executed.
- The e-cash register has called the Hello service and received a successful response.
- After a successful re-personalization process, the e-cash register can open a new fiscal day.
- The numbering of receipts and reports (recordCounter) continues without duplication or gaps.

If the re-personalization process fails, it can be reattempted by calling the re-personalization service again, followed by invoking the services listed above

5.14.2 Technical description of the service

The re-personalization of the e-cash register is initiated with the "ownerChange" service.

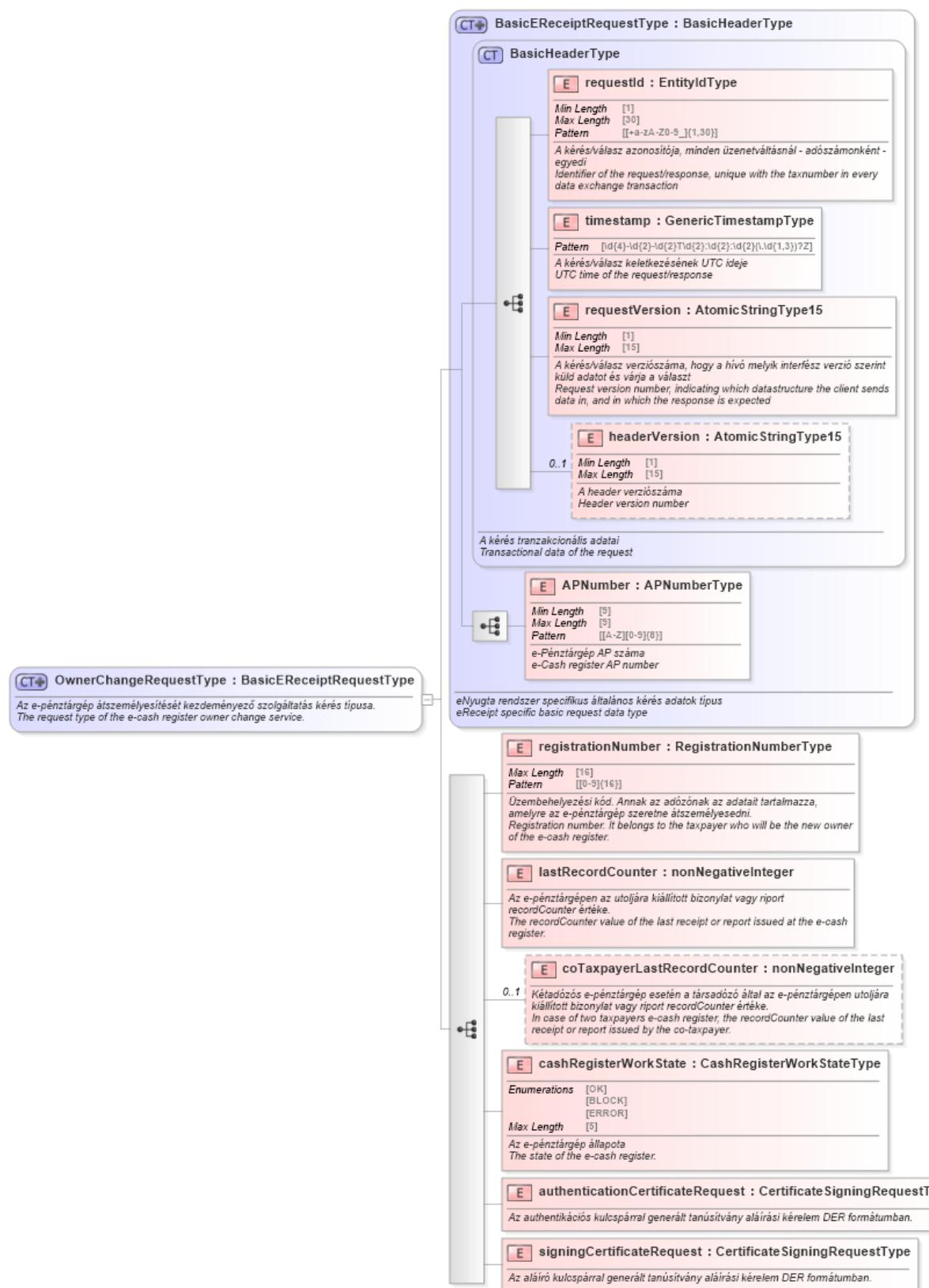
- Context root: /eReceiptMgmt/v1
- URL: /ownerChange



-
- Request object: OwnerChangeRequest The technical description of the request object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
 - Response object: OwnerChangeResponse The technical description of the response object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.



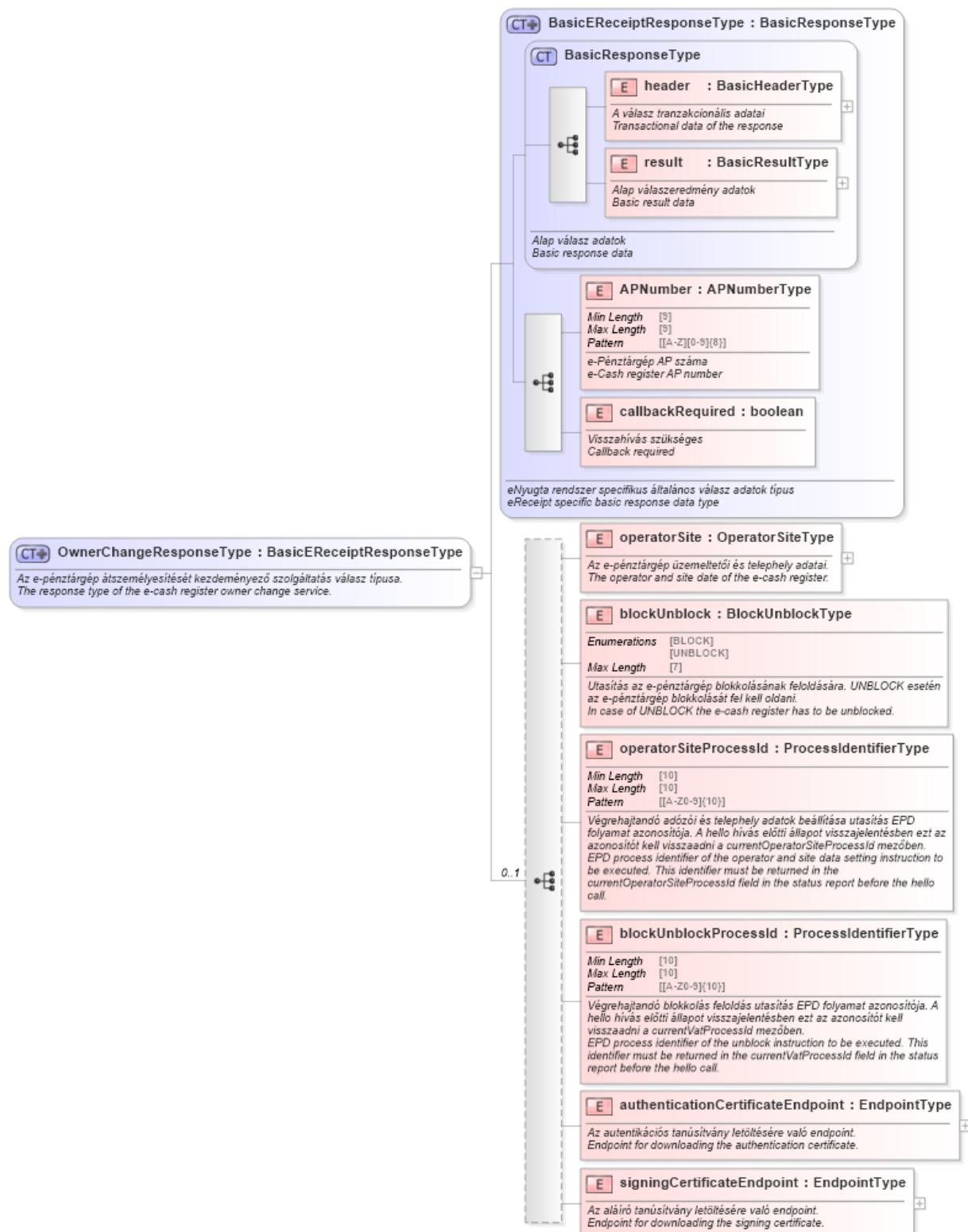
The OwnerChangeRequest request object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201



The OwnerChangeResponse response object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201



5.15 Hello service

5.15.1 Business description of the service

After successful registration (initial commissioning) or re-personalization, the e-cash register calls this service to notify NAV-I that the registration or re-personalization process has been completed. Following registration or re-personalization, the e-cash register is capable of issuing receipts as per normal operation.

The Hello message must confirm the recording of operator data and the new VAT structure (process identifier). In the case of re-personalization, only confirmation of the operator data update needs to be submitted.

The e-cash register is considered commissioned after successfully calling this service.

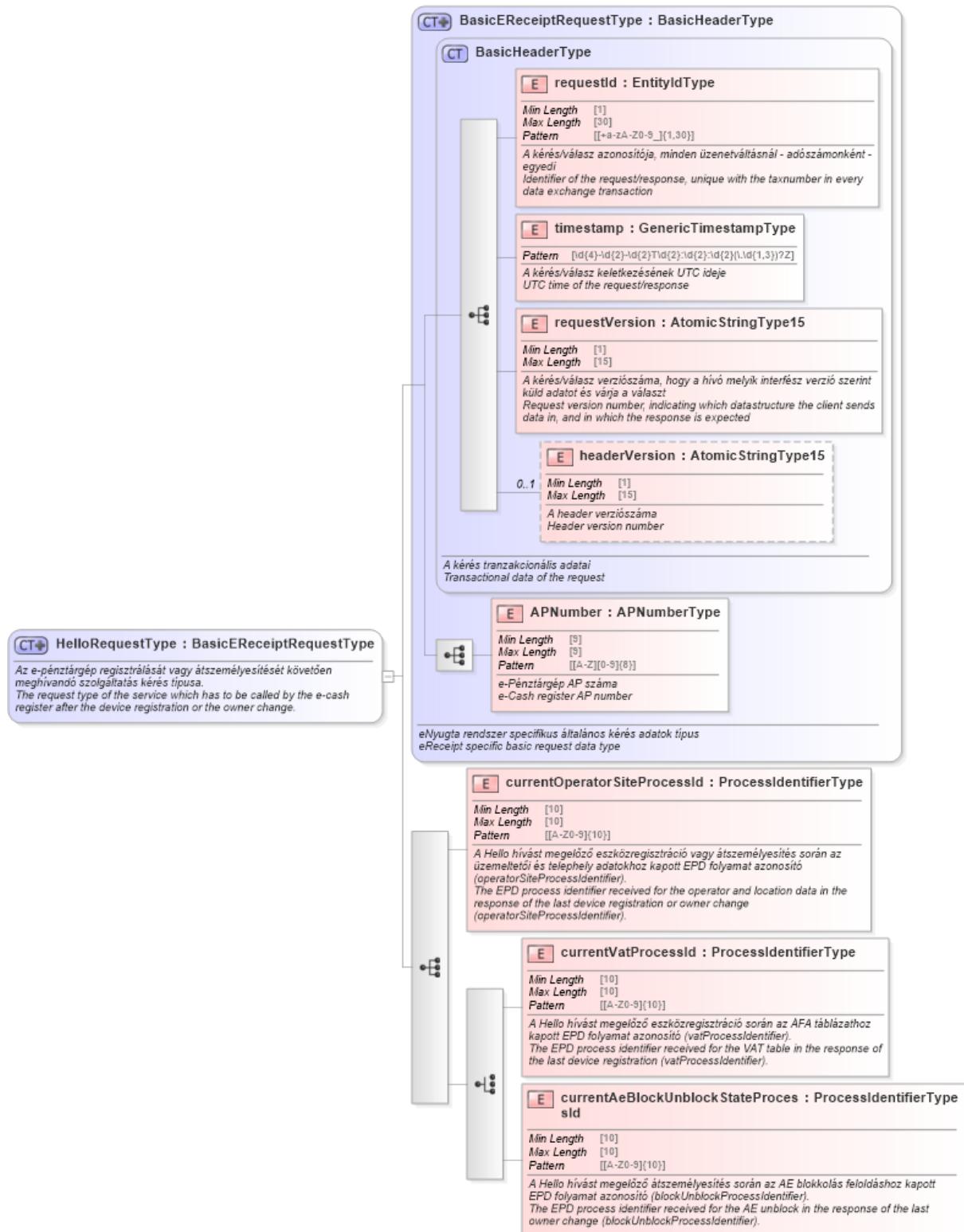
5.15.2 Technical description of the service

If the e-cash register has completed the registration or re-personalization process, it must signal this by calling the "hello" service.

- Context root: /eReceiptMgmt/v1
- URL: /hello
- Request object: HelloRequest. The technical description of the request object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- Response object: HelloResponse. The technical description of the response object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.



The HelloRequest request object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

The HelloResponse response object does not contain endpoint-specific data beyond the header and result data.



5.16 Certificate Renewal

The e-cash registers possess two certificates: the authentication certificate used for identifying the devices in the NAV-I central system, and the signing certificate used for verifying the authenticity of the transmitted data. These two types of certificates must be managed independently within the e-cash registers. The renewal must be completed no earlier than 30 days before the certificate's expiration. This service ensures the availability of the renewed certificate for download. The download or certificate replacement is also possible after the existing certificate expires. During the service request, a verification is performed to ensure that no more than 30 days remain until the certificate's expiration. If there is more time remaining, the certificate renewal request is rejected.

5.16.1 Business description of the service

The e-cash register can access central system services using the authentication certificate. The authenticity of the transmitted data is ensured through digital signing using the signing certificate, which is verified by the NAV-I system in every case. Certificates issued via NAV-I are valid for two years plus 30 days, and they must be renewed before expiration. This service allows for the submission of renewal requests for these certificates.

The renewal process can be initiated automatically by the e-cash register before certificate expiration by submitting a newly generated Certificate Signing Request (CSR) in CMS SignedData format (where the new CSR envelope is signed with the old certificate). The e-cash register must also be capable of invoking this service manually upon operator intervention. The tax unit is responsible for securely storing the renewed certificate obtained from NAV-I. The authentication certificate must be placed in a password-protected key store.

In the case of an expired certificate, the e-cash register operator must request a certificate renewal code from the KOBÁK Portal, which must be submitted when invoking the service. In this case, the request must be submitted in CSR format without CMS SignedData packaging, as the e-cash register can no longer sign the CMS with a valid certificate.

The service response returns a URL from which the newly issued certificate can be downloaded. The e-cash register must wait 5 seconds before attempting the first download. If the certificate is not yet available at the provided URL, the retry attempts must be spaced at least 10 seconds apart. If the certificate is still not available within 2 minutes from receiving the URL, the renewal is considered unsuccessful, and the service must be re-invoked with a new CSR (new keys).

In the case of a revoked certificate, the operator must also be provided with the ability to obtain a new certificate, following the same process as for an expired certificate renewal.

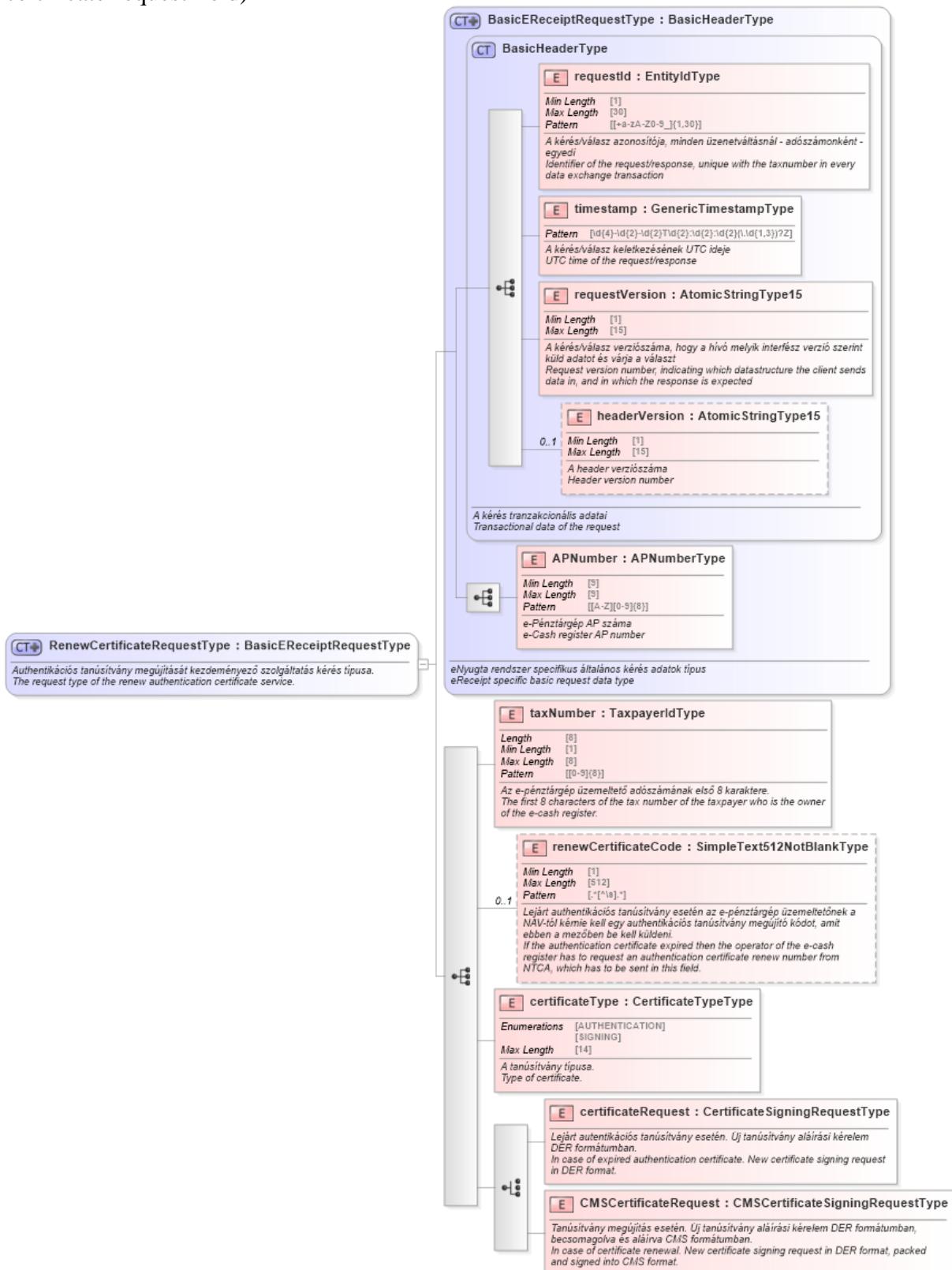
5.16.2 Technical description of the service

The certificate renewal of the e-cash register is performed using the "renewCertificate" service.

- Context root: /eReceiptMgmt/v1
- URL: /renewCertificate
- Request object: RenewCertificateRequest. The technical description of the service request object is found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- Response object: RenewCertificateResponse. The technical description of the service response object is found in the "[Description of business data content \(XSD model types and elements\)](#)" section.

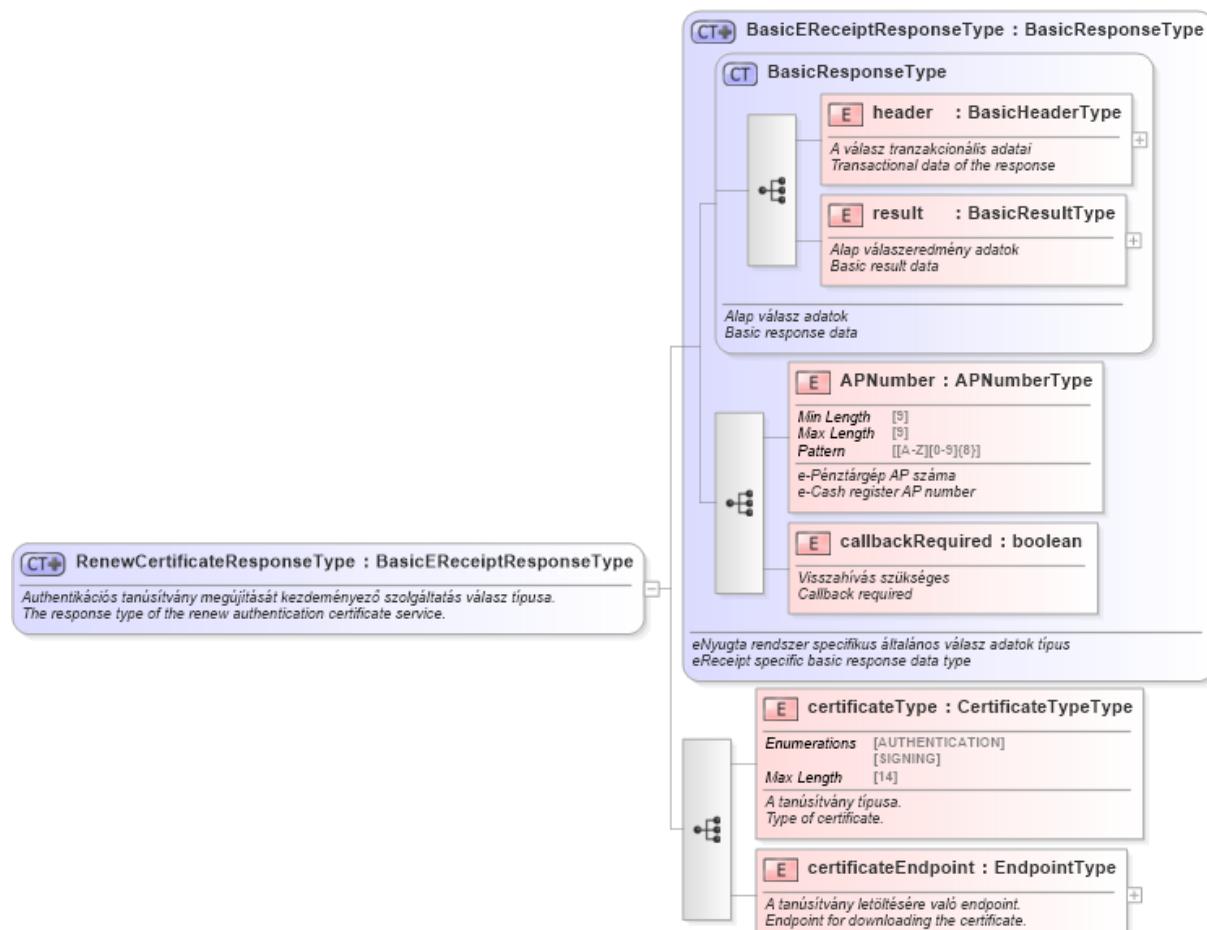


For pre-expiration renewal, the optional renewCertificateCode field in the RenewCertificateRequest must not be filled. The CSR must be wrapped in a CMS envelope and submitted in the CMSCertificateRequest field (the request must not contain the certificateRequest field)





RenewCertificateResponse response object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

The renewal of the e-cash register certificate in case of an expired certificate is performed by the "renewExpiredCertificate" service. This service can be called without an authentication certificate.

- Context root: /eReceiptMgmt/v1
- URL: /renewExpiredCertificate
- **Request object:** RenewCertificateRequest. The technical description of the request object for the service is found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- Response object: RenewCertificateResponse. The technical description of the response object for the service is found in the "[Description of business data content \(XSD model types and elements\)](#)" section.

The endpoint for expired certificate renewal uses the same request and response objects as the pre-expiry renewal service, but their filling differs.

For an expired certificate, the renewCertificateCode field in the **RenewCertificateRequest** must be filled in, and the CSR must be submitted in the certificateRequest field (the request must not contain the CMSCertificateRequest field).



5.17 Product catalog query

5.17.1 Business description of the service

With the help of the service, queries can be executed in the global trade item number database through NAV-I.

The e-cash register can request the product catalog from NAV-I by calling this service. Currently, the service supports one search mode, product identifier – for example, a barcode read from the product packaging – can be used to query the product data. The code can be EAN, GS1, or any identifier displayed with one- or two-dimensional barcodes or written with letters and digits. At least 5 (five) alphanumeric characters must be provided for the search.

The service attempts to match the identifier provided in the search parameter against all product identifier types stored in the global trade database.

The search may result in multiple matches depending on the type of identifier used in the query.

It only provides a result in case of an exact match (for example, when "123456" is given, it will not return the product associated with the "1234567890123" GS1 code).

If the product identifier is found in the database, the service returns the following data:

- DTSZK identification number
- Identifier indicated on the product (e.g. EAN 13- or 8-character, internal identifier)
- Product name, description
- Manufacturer (if known)
- Validity start date
- Validity end date
- Status
- Customs Tariff Number
- Category
- Unit of measure (piece, kg, liter, etc.)

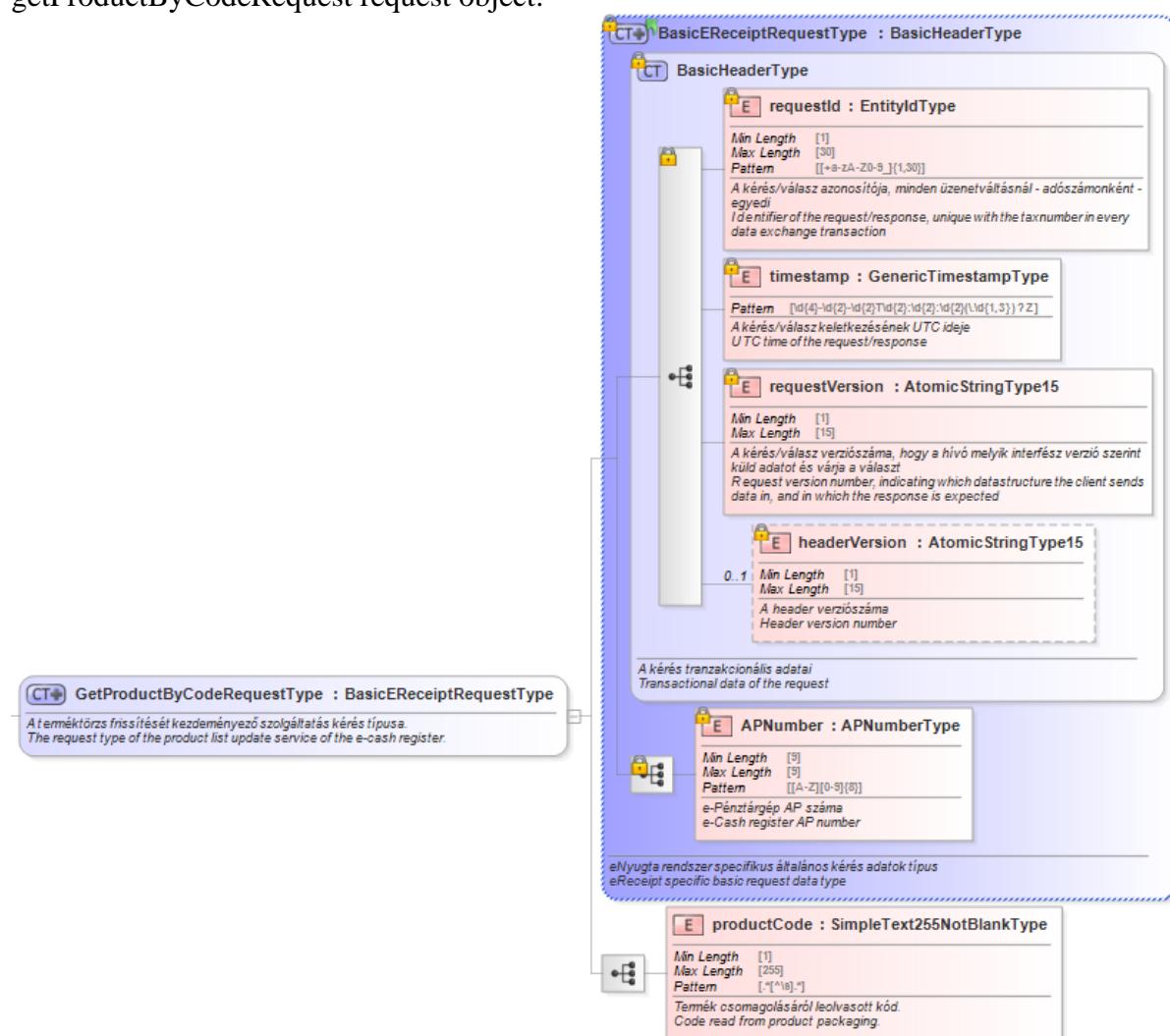
5.17.2 Technical description of the service

The initiation of the product catalog download is performed by the "productList" service.

- Context root: /eReceiptMgmt/v1
- URL: dpsc/getProductByCode
- **Request object:** getProductByCodeRequest. The technical description of the request object for the service is found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- **Response object:** getProductByCodeResponse. The technical description of the response object for the service is found in the "[Description of business data content \(XSD model types and elements\)](#)" section.

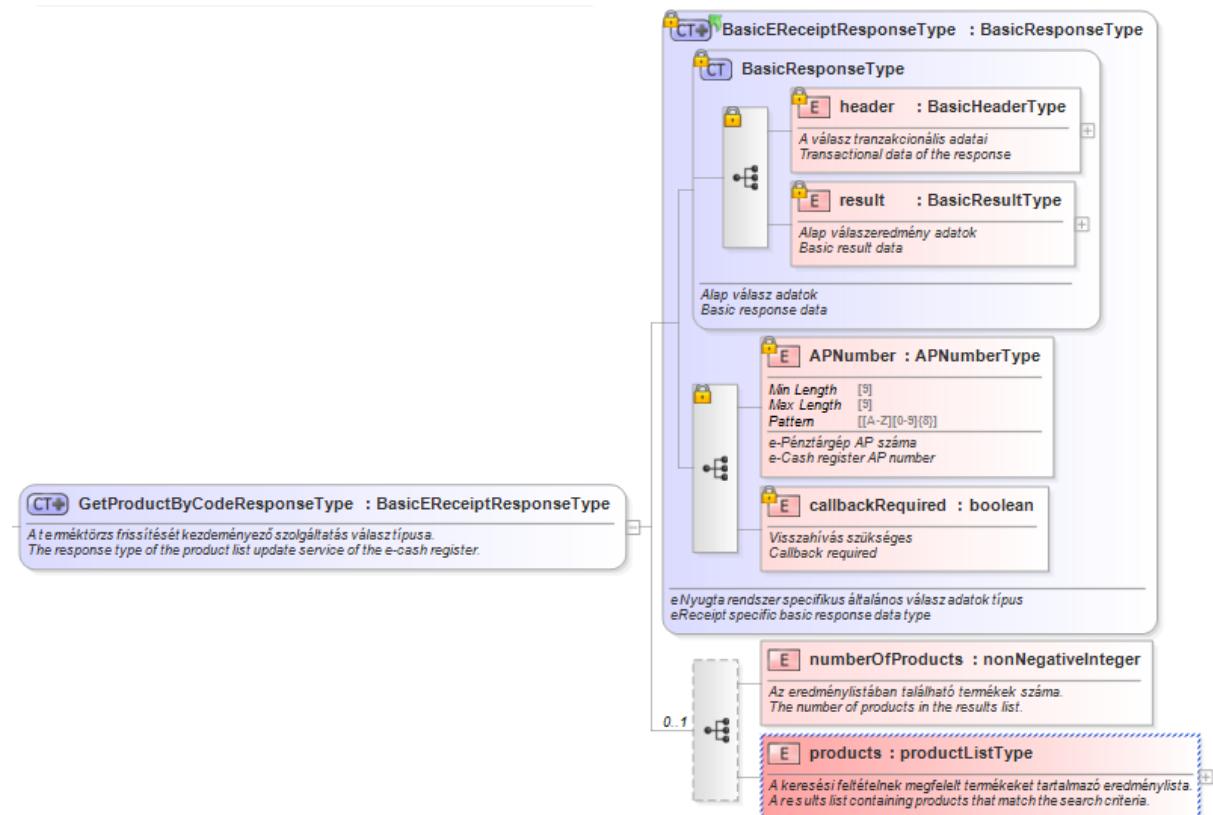


getProductByCodeRequest request object:

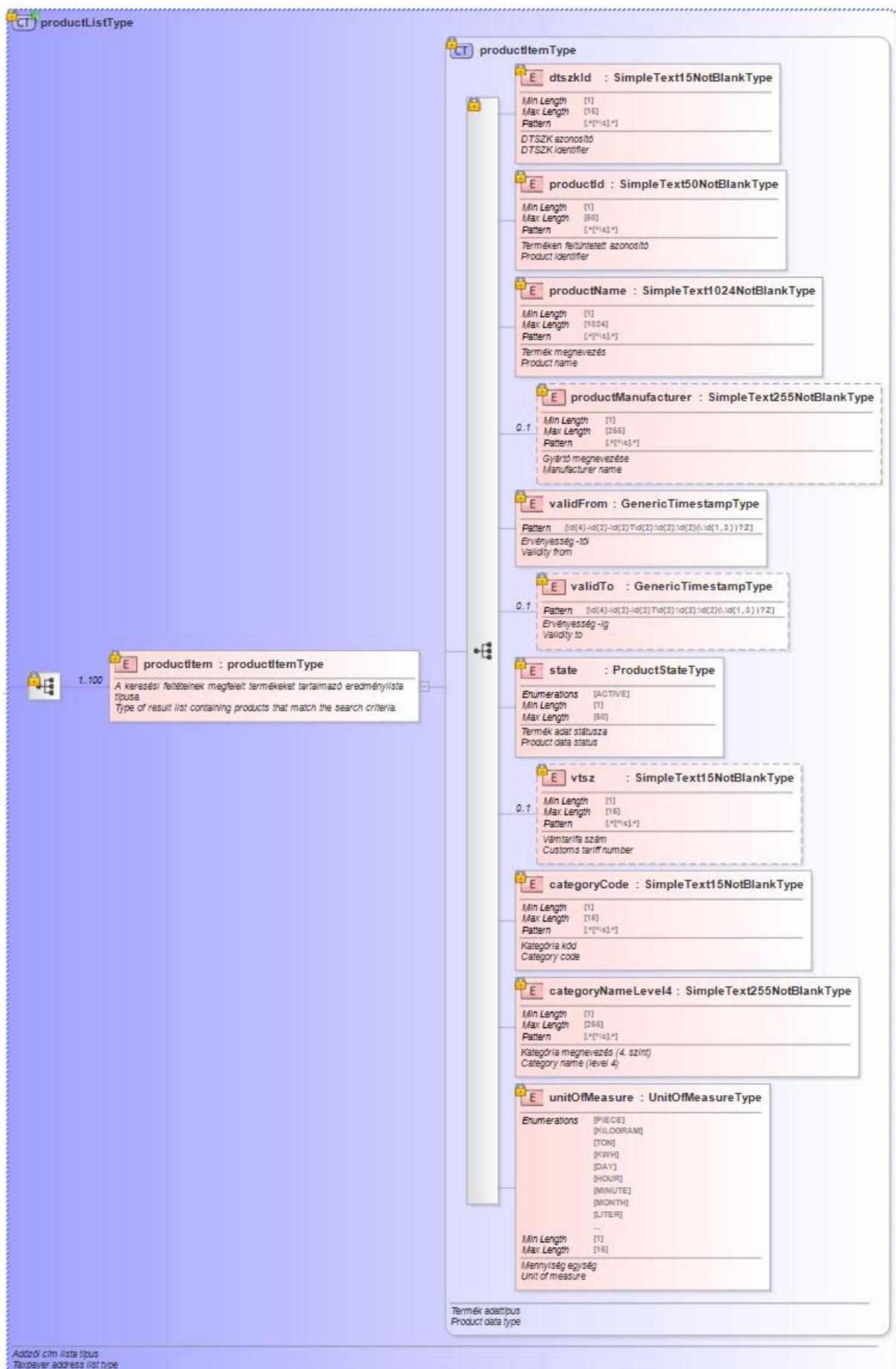




getProductByCodeResponse response object:



Where the `ProductListType`:





5.18 Request for Submission of Missing Documents

5.18.1 Business description of the service

Through the Communication Manager service, the NAV-I can request the e-cash register to submit missing or resubmitted receipts or reports, providing the recordCounter value of the document and a unique process identifier.

The submission of the receipt or report takes place through the standard submission endpoint. However, documents or reports submitted in response to this request must include the process identifier in the request.

The e-cash register's tax unit must submit the requested document or report using the current XML schema version applicable for standard data submission, even if the document was initially recorded using a different schema version at the time of its original submission.

5.18.2 Technical description of the service

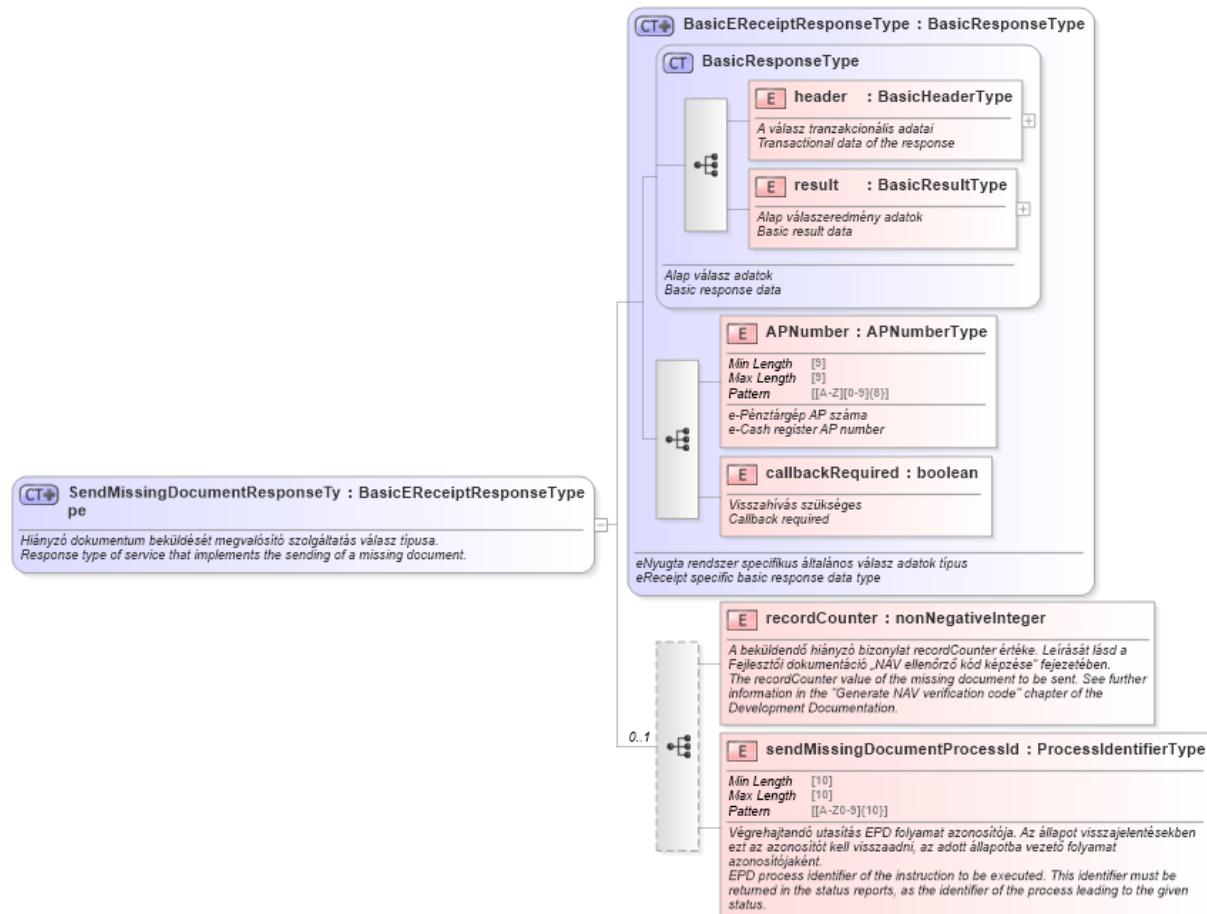
The identification of the missing or resubmitted document is handled by the "sendMissingDocument" service.

- Context root: /eReceiptMgmt/v1
- URL: /sendMissingDocument
- Request object: SendMissingDocumentRequest – The technical description of this request object is found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.
- Response object: SendMissingDocumentResponse – The technical description of this response object is found in the "[Description of business data content \(XSD model types and elements\)](#)" chapter.

The SendMissingDocumentRequest request object only contains the header data described in the "[General Technical Data](#)" chapter.



SendMissingDocumentResponse response object:



Liquid Studio - Developer Bundle (Trial) 20.7.17.13201

6 Business services provided by mobile service providers/cloud service providers

6.1 Instruction for immediate login

NAV may send instructions via the mobile service provider or cloud service provider for the e-cash register to initiate an immediate connection with the NAV-I central system.

The instruction is initiated by NAV-I towards the mobile or cloud service provider, which must immediately forward it to the e-cash register.

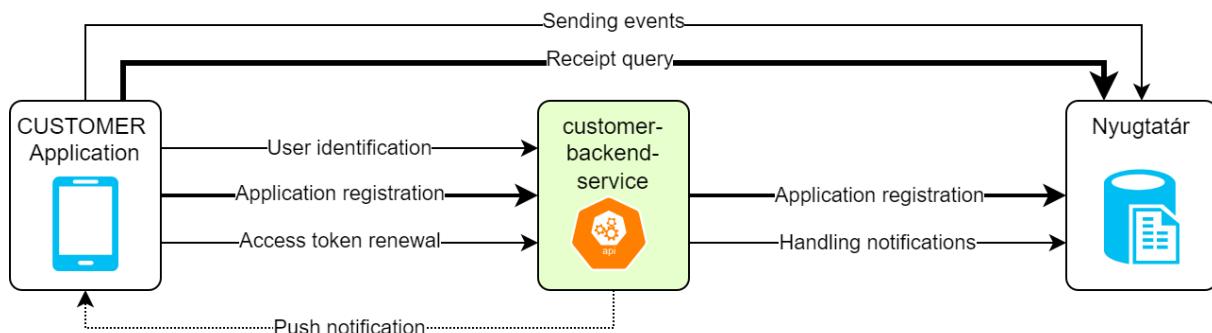
A hardware-based e-cash register must be capable of receiving an SMS from the mobile service provider and based on the message received, must call the Communication Manager service on the NAV central server. The description of the service can be found in the "[Communication Manager](#)" section.

For a cloud-based e-cash register, the cloud service provider receiving the instruction must call the Communication Manager service on the NAV central server using the AP number of the e-cash register received from NAV.

7 Customer Application Requirements

The customer application must have its own backend service, which must have at least one function that assists in carrying out the application registration in the receipt store.

The backend service may be extended with additional functions, if they do not conflict with the requirements described in the Decree.



The calling of the receipt store registration notification query endpoints can only be performed from the customer backend. The receipt store verifies the backend connection with two factors:

- The backend service may only connect from a fixed IP address, which must be provided during the application authorization.
- The customer backend presents a client certificate to the receipt store. If a customer backend serves multiple applications with different authorization numbers (e.g. iOS and Android versions), a dedicated backend authentication certificate must be registered for each authorization number, which must be handled separately in the backend, and the



data belonging to different authorization numbers may not be mixed in the receipt store calls.

The customer application may optionally use push messaging to notify users when the receipt is downloaded. The customer backend can download the notification list from the receipt store through the endpoints described in the subsections [Retrieving notification data](#) and [Retrieving notification download workflows](#).

The receipt query service can only be called from the application running on the user's mobile device, not from the customer backends.

7.1 Customer Application Registration

Before utilizing the services listed in the "[Services provided by the Receipt Repository](#)" section, the customer application must complete a registration as per described in "[Services provided by the Receipt Repository](#)" chapter. The registration must be performed using the service described in the [Customer Application Registration](#).

The registration of the application can only be carried out by the customer backend service. There is no technical restriction on the registration of the application user in the backend service, the application developer may provide named or anonymous user accounts, and may use an authentication service offered by a third party.

A unique UUID type identifier must be assigned to the user, which must be provided to the receipt store in the request data structure during the application registration.

During registration, the receipt store returns a so-called "JWT access token." This must be used in the application for receipt queries.

This token must be obligatorily forwarded to the application by the backend.

The expiration time of the token is 8 hours, and a new token can be requested no earlier than 30 minutes before expiration.

The tokens received during registration are always associated with a specific software version; if the user downloads a new version, registration must also be completed for that version.

For every Receipt Store service call, the identifier token received during the process described in the subsection [Registering the customer application](#) must be provided as described in the [HTTP headers](#) section.

7.2 Key Generation

Upon user request, the customer application – for the purpose of eReceipt encryption – must generate an ECC key pair for data encryption. The encryption key pair must be generated according to the ECC SECP256R1 curve (RFC5480).

The customer application must ensure the secure (encrypted) storage of the generated key pair, as it will be used for decrypting encrypted data and retrieving receipts from the receipt repository. If the key is lost, the user will not be able to retrieve and decrypt the data stored in the receipt repository.



7.3 Receipt Download

The customer application must be capable of downloading receipts transmitted to the receipt repository by e-cash registers and displaying them in a readable and interpretable format for the user.

If the customer application provides a QR code to the e-cash register, it must store the time of QR code generation and the uniquely generated encryption key. If the customer application did not provide a QR code or the QR code did not contain these details, it must be able to interpret the QR code on a printed copy of the e-receipt and store the search date, encryption key, and other data contained therein.

From the output QR code of the e-cash register, the customer application must be able to clearly determine which party generated the key pair for the given receipt, and this information must be made available to the customer. This makes it possible to indicate situations where the customer reads an output QR code from a paper copy for which their own application did not provide the encryption key.

After the purchase, the customer application must call the "Receipt Download" service using the stored search date, the raw SHA-256 hash value of the "compressed" public key of the encryption key pair (the search key), and the QR code data generated by the e-cash register (excluding the decryption key). The receipt must then be downloaded from the receipt repository.

The customer applications, in case of unsuccessful download, may attempt to download the receipt at the intervals described in the "Receipt query" service.

Downloaded receipts must be stored encrypted with a symmetric key on the device, or in the central database for customer applications. The receipts must then be decrypted as described in the "[Receipt Decryption](#)" section and displayed in a readable and interpretable format for the user.

If an old receipt is queried using the "Receipt Download" service, for example, as part of a Recovery function, an archive key will be returned instead of the receipt. This key must be used to call the "Archived Receipt Download" service.

7.4 Receipt Decryption

The customer application must be able to decrypt the encrypted data in the downloaded receipt attachments using the private key of the encryption certificate.

The customer application must decrypt the receipt data received using the ECIES algorithm. The encryption details are found in the "[Encryption](#)" section. The encrypted data can be decrypted using the customer's private encryption key pair. If the customer application was not used during the purchase, the e-cash register generates the customer's encryption key pair and prints it onto the QR code on the receipt copy.



7.5 Export

The customer application must be capable of exporting the downloaded electronic receipts in a format that can be saved for further use by the user. The export must be possible in two ways:

- A human-readable format (HTML, Excel)
- A format readable by another customer application

The machine-readable export format for customer applications is standardized. The export format is described in eReceiptExport.xsd. The exported data must only be stored in an encrypted package using either password protection or an asymmetrically derived cryptographic key (e.g., ECIES) with the AES-256 algorithm.

7.6 Saving Receipt Header Data for Recovery from the Receipt Repository

The customer application must be able to automatically or manually save the encryption certificate pair, the dates and search keys required for receipt lookup, and, for receipts downloaded via QR code scanning, the QR code fields (including the signature and certificate serial number) to ensure that receipts remain accessible if the device or customer application is changed.

In addition to the above, the following data must also be saved:

- Receipt serial number
- Receipt timestamp
- Gross amount of the receipt
- Seller's name (optional)

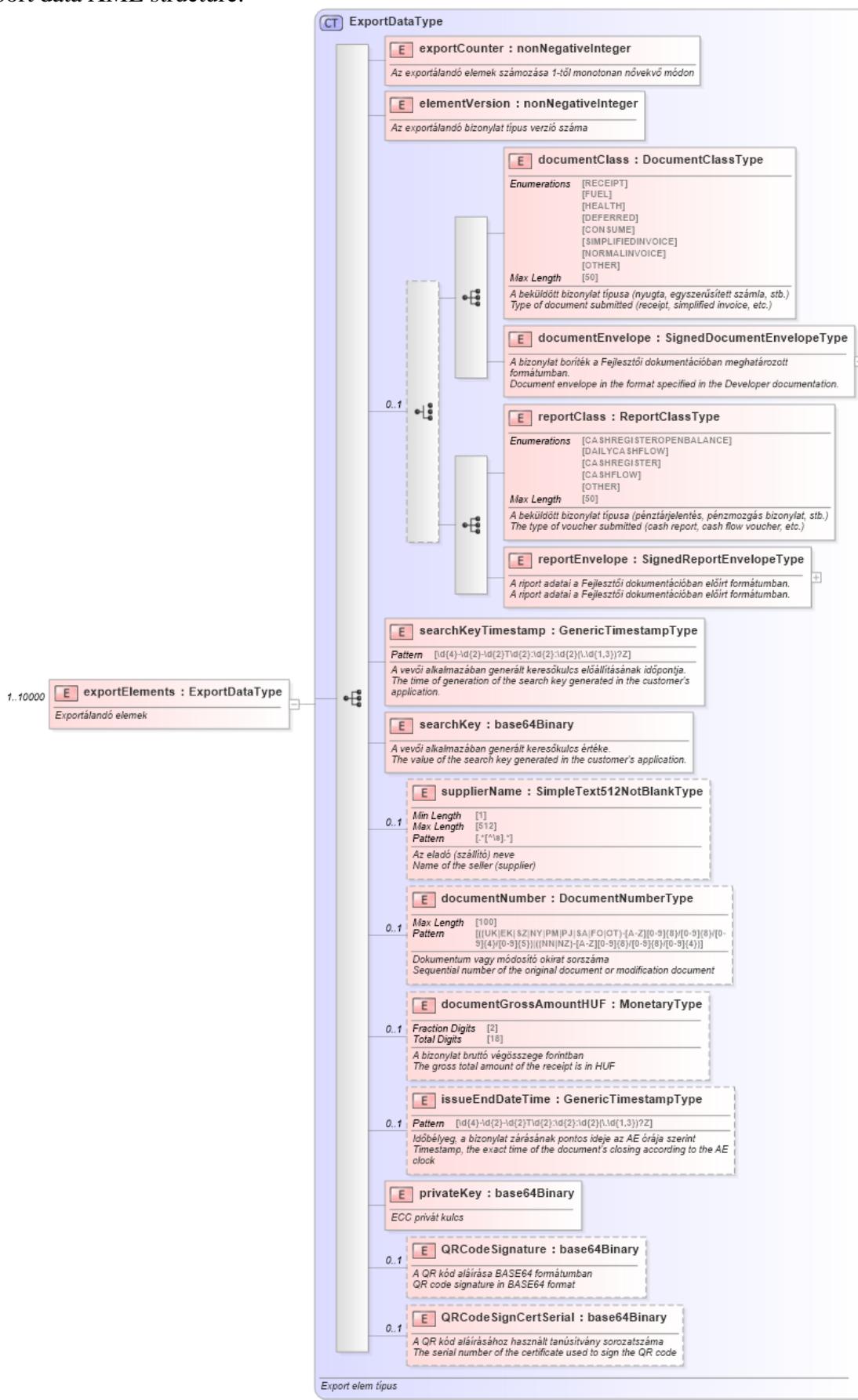
Optionally, the application may save the entire receipt envelope downloaded from the receipt repository along with the above data.

The encryption certificate pair must be stored in PEM format, and the certificate must be password-protected.

Saved data must be stored in XML format, as described in eReceiptExport.xsd. The recovery data must only be stored in a protected package encrypted with the AES-256 algorithm.



Export data XML structure:





7.7 Import

The customer application must be capable of importing the data content of a package exported from another or the same customer application according to eReceiptExport.xsd.

7.8 Recovery

The customer application must ensure the receipt recovery function in case of device or customer application replacement.

The recovery can be performed using either a locally saved full export or the receipt repository. The application must load the certificate pair and receipt data files saved by the export functions ("Export" section). After loading, the application must query the receipt data from the repository using the ["Receipt Download"](#) Receipt Download function.

7.9 QR code generation

The customer application must be capable of generating a QR code containing the data required for issuing a receipt or simplified invoice.

Before the purchase begins, the customer application must be able to generate a random encryption key pair upon user interaction. The generated customer's public encryption key must be passed to the e-cash register via the QR code. The application must store the encryption key and the QR code generation date, enabling it to download the receipt from the receipt repository. The QR code requirements are described in the "[Generating input QR code by E-cash register](#)" section.

7.10 Version control

The customer application must warn the user to update in case the version being run has been revoked by the NAV. The receipt store rejects the application registration endpoint call made with revoked versions (through the customer backend), indicating in the response to the customer backend that the token issuance was rejected due to the revocation of the version. Based on this, the customer backend can also provide a clear indication to the client application.

Revoked application versions may not read output QR codes and may not provide a key to the e-cash register, that is, the scanning and customer QR code display functions must be disabled until the update. The display of already downloaded receipts and the data export functions must continue to work, leaving the possibility to view and migrate data from applications that the distributor may not intend to update.

7.11 Handling of inspection and live modes

Since the authorization applies to a specific build of the application, every build must be able to call both environments, but the method or tool for switching to the inspection environment must not be offered to the user.

Since the receipt store rejects the application registration request initiated with non-authorized versions, these software versions may only call the inspection environment. The application must always first call the application registration on the live customer backend, and the authorized / non-authorized status is indicated back to the application by the live customer backend. If the given version is not authorized, the application must switch to the inspection



environment. This switch is one-way, and cannot be switched back to live even if the version is later authorized. To switch back to live, the application (and its data) must be deleted from the given device, and after reinstallation it can connect again to the live backend.

A customer application operating in the live environment with authorization must be able to be switched to the inspection environment by scanning a special QR code specifically designed for this purpose. The definition of the QR code content and the exclusion of the possibility of accidental switching is the responsibility of the application developer. The developer must provide the switching QR code, or in the case of a dynamic code its generation method, to the NAV. The redirection using the QR code is also one-way, and switching back likewise requires the deletion of the application and its data.

8 Services Provided by the Receipt Repository

The receipt store endpoints that can be called directly by the customer application or by the customer backend are available under different DNS names, see the chapter "["Accessiblity of the environments."](#)"

8.1 Technical Description of Services

8.1.1 General Technical Data

All messages exchanged between customer applications and the Receipt Repository must be encoded using the UTF-8-character set. Messages must be sent over an SSL channel using the HTTP/POST method to the central system.

The service endpoint context root can be the following:

- /eDocumentStore/v1 – Endpoints accessible by the customer application via the Receipt Repository.

The technical description of the service endpoints described in this section includes the context root of the respective endpoint.

If the contact initiated by the customer application fails, the communication can be repeated at the earliest five seconds after the previous communication, with a maximum of two attempts.

If these attempts remain unsuccessful, the customer application may attempt to contact the Receipt Repository again 30 minutes after the last failed call.

8.1.2 HTTP headers

For every service invoked with the POST method, the following HTTP header fields must be provided:

- content-type=application/xml ;charset=UTF-8
- accept=application/xml

The JWT access token is provided in the identification header parameter (there is a space and not a new line between the word “Bearer” and the token):

Authorization: Bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJKYzg1Y2M2ZS1hYjE3LTR1OGItOTAwNy03ZWU4ZTRjYWFjNmYiLCJpYXQiOjE3MTIzMjNDUwMDAsImV4cCI6MTcxMjM0ODYwM CwiaXNzIjoibmF2Lmdvdi5odSIsImF1ZCI6ImN1c3RvbWVyX2JhY2t1bmRfaWQifQ.EaqTB s7Q9q3BSjepYreUOGHjvj_4vdTn7JUSK1TZf4



The following parameter must be provided only for endpoints accessible from the application running on the customer's mobile device:

- User-Agent=<app neve>/<app verzió> (<os>; <os verzió>; <eszköz>)<platform>/<platformversion>

When calling the Nyugtatár, the customer application sends its application- and environment-specific data in the User-Agent HTTP header. The header must be dynamically generated from the following data:

- Application data:
 - Application name – The name registered in the app store
 - Application version – The version number registered in the app store
- Environmental data:
 - Operating system, "iOS" or "Android"
 - Operating system version number (e.g., Build.VERSION.RELEASE on Android, UIDevice.current.systemVersion on iOS)
 - Device – The model identifier of the device running the application (e.g., Build.MODEL on Android, UIDevice.current.model on iOS)
- Platform data:
 - Platform – The official name of the application's development framework.
 - Platform version – The version of the framework in which the specific application version was created.

Important! The User-Agent HTTP header may not contain accented characters, therefore, for example, accents must be removed from the application name before submission.

The User-Agent HTTP header structure follows the format:

<app name>/<app version> (<os>; <os version>; <device>) <platform>/<platform version>

Example: "NAV Vevoi App/1.0 (iOS; 18.0; iPhone13,1) ExoPlayerLib/2.11.4", or "NAV Vevoi App/1.0 (Android; 14.0; Pixel 7a) ExoPlayerLib/2.9.0"

8.1.3 HTTP status codes

The service always responds with HTTP 200 when the request is valid. This does not necessarily indicate that the business execution of the request was successful, only that the request was correctly formatted from an IT perspective, and the invoked resource could read and process it. In other words, it met the request format specified in the service's technical description, and the message from the invoked service conformed to the schema describing the service call (XSD).

In case of an incorrect request or other technical error, the returned results are detailed in the error code table in the ["Error Handling"](#) section.

8.1.4 Response time, timeout

The server typically serves requests with response times under 500 ms. The blocking timeout for synchronous calls is 5000 ms. On the client side, response times exceeding this value can be treated as a timeout.



8.2 Customer application registration

A service for registering the customer application on the Nyugtatár side.

8.2.1 Business description of the service

After downloading the customer application from the application store, it must be registered in the Nyugtatár system before its first use. During registration, the software version details of the customer application must be provided. Based on the software data provided, NAV verifies whether the given version has been inspected, whether its use is permitted, and whether downloading a newer version is necessary.

If the applied version is acceptable, the Nyugtatár system returns a JWT token at the end of the registration. This return token must be transmitted during every subsequent service call when using the receipt storage services.

If any error occurs during the registration process, the Nyugtatár system will notify the application according to the error handling section described in the documentation.

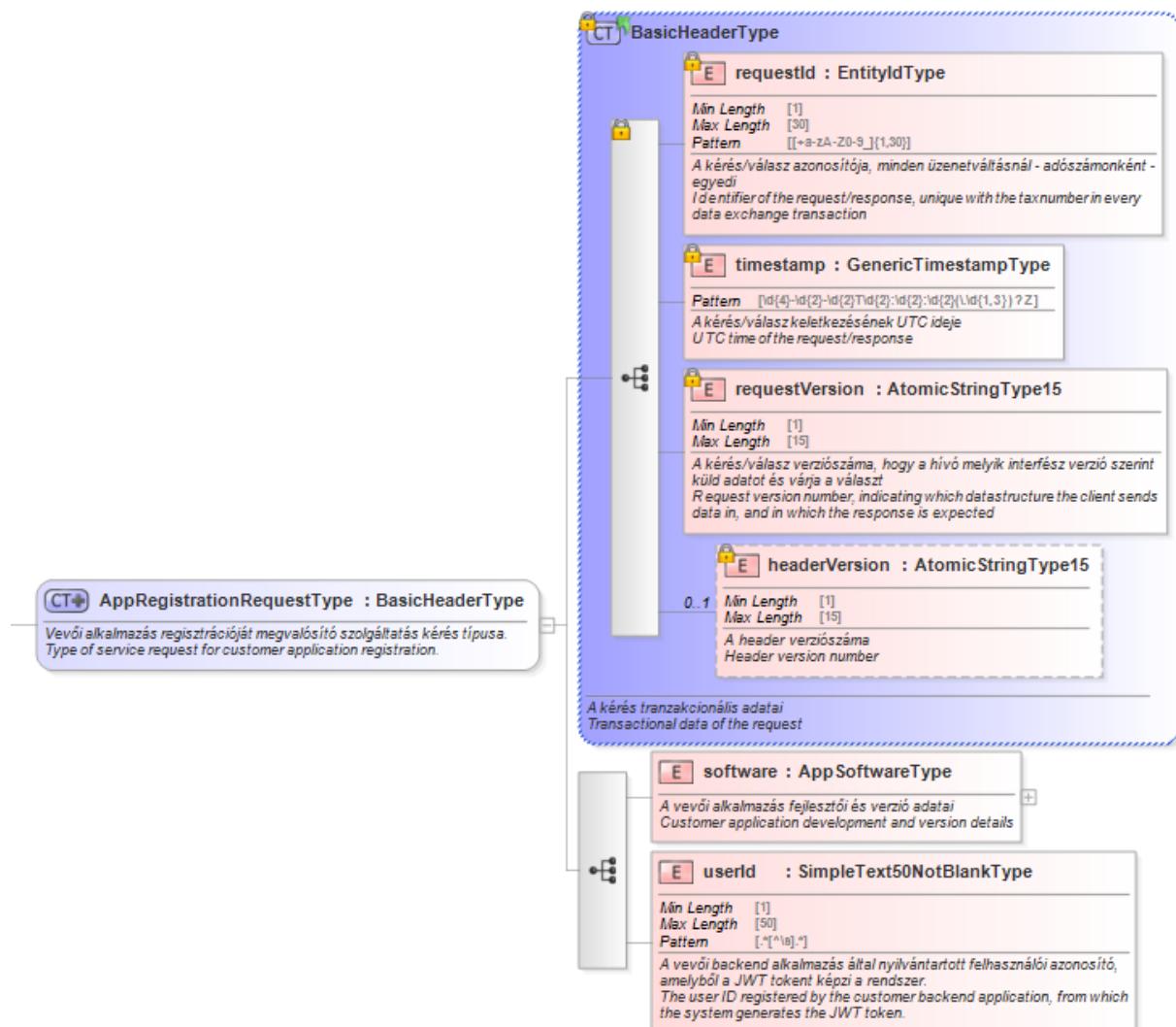
8.2.2 Technical description of the service

The registration of the customer application provided by the receipt storage system is implemented via the "appRegistration" service.

- Context root: /DocStoreReg/v1
- URL: /appRegistration
- Request object: AppRegistrationRequest. The technological description of the service's request object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section. Description of business data content (XSD model types and elements)
- Response object: AppRegistrationResponse. The technological description of the service's response object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.

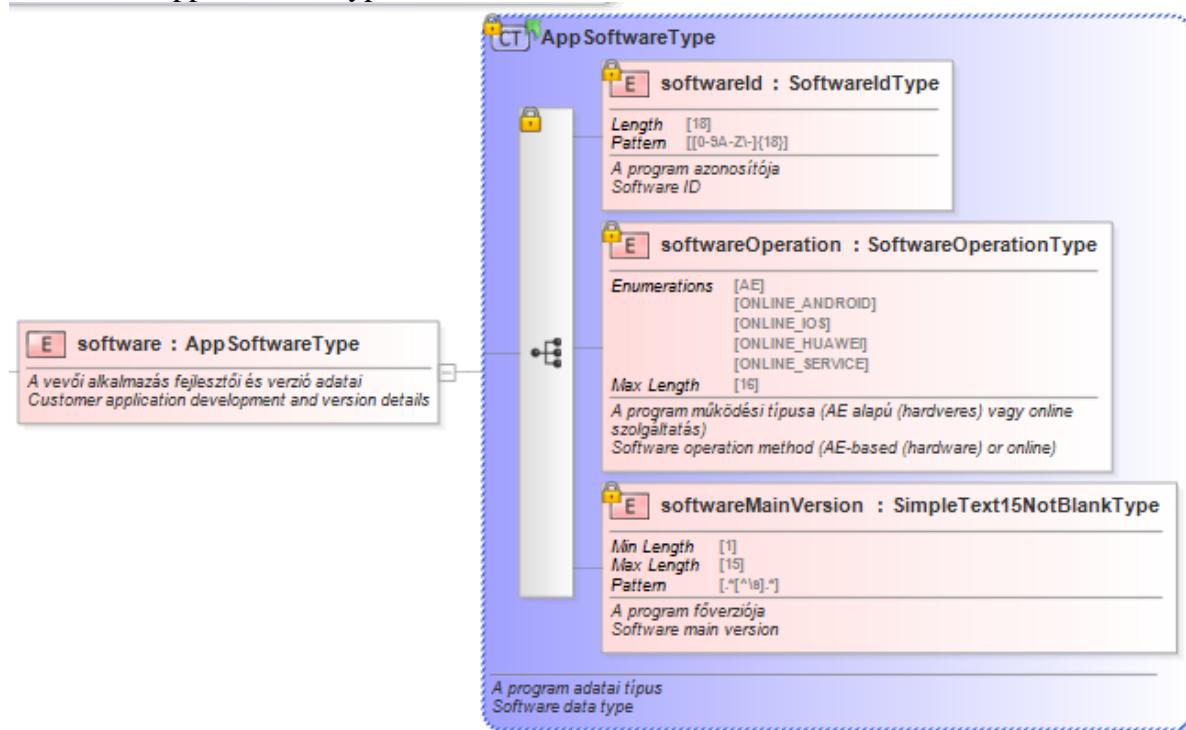


AppRegistration Request object:



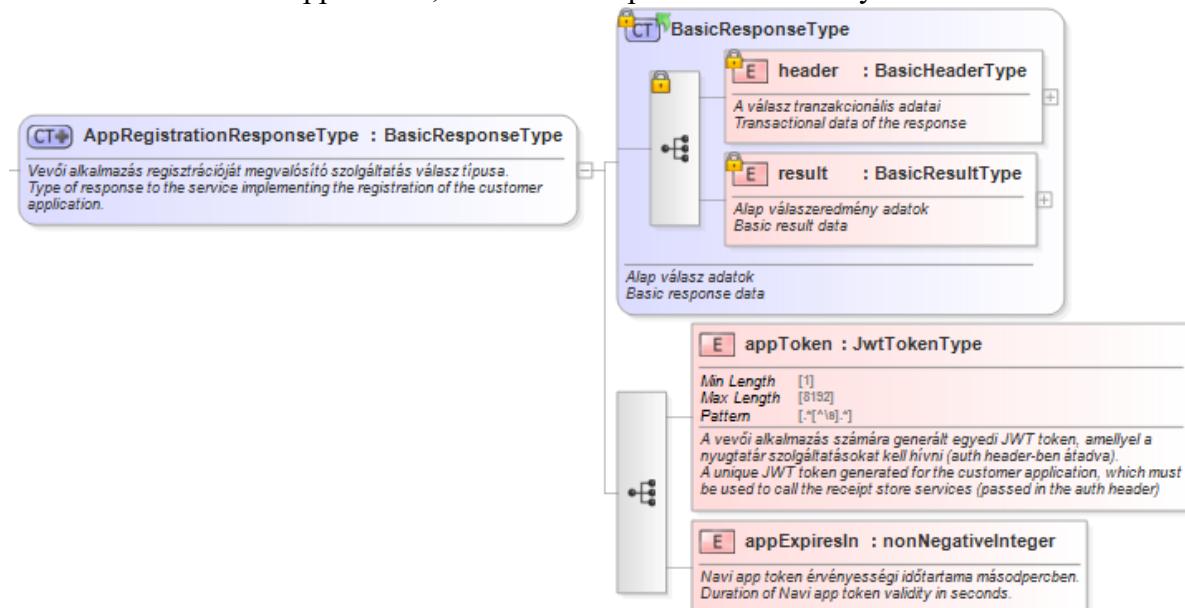


Where the AppSoftware Type:



AppRegistrationResponseType object:

In case of a customer application, the softwareOperation enum may not take the value “AE”.





8.3 Receipt query

Receipts submitted by electronic cash registers (e-cash registers) that also concern the customer are stored in the receipt repository and will be available for the period specified by the regulation. Customers can retrieve their receipts from the receipt repository.

8.3.1 Business description of service

The purpose of this service is to provide receipts to customer applications as efficiently as possible and with an appropriate response time. The service can be used without user authentication.

The receipt query is always performed using a search date and a search key:

- If the customer application provided a QR code to the e-cash register during the purchase, which contained the encryption key, the search parameters are as follows:
 - Search date: The timestamp when the QR code was generated.
 - Search key: The SHA-256 value of the customer encryption (raw, "compressed") public key contained in the transmitted QR code.
- If the customer application did not provide a QR code to the e-cash register during the purchase, or the QR code did not contain the customer encryption key, the search parameters are extracted from the QR code on the printed receipt copy:
 - Search date: The receipt issuance timestamp with second precision.
 - Search key: The PSHA-256 hash value of the raw, "compressed" public key of the customer encryption key pair generated by the e-cash register.
 - Receipt serial number.
 - Total amount of the receipt in HUF.
 - QR code signature: The base64 character sequence copied from the QR code.
 - Serial number of the certificate used for QR code signing, copied as a base64 character sequence from the code.

If the receipt is not yet available in the receipt repository at the time of the query, the next query must wait at least 5 seconds, and the query may be repeated up to three times. After that, the customer application must wait at least 5 minutes before making another query for the receipt. Meanwhile, other receipts can be queried by the customer application.

If the receipt is available in the fast-access storage of the receipt repository, the service immediately returns the receipt to the customer application. If an old receipt is queried, the response does not contain the receipt itself but an acknowledgment of the query's acceptance. After this, the receipt store loads the receipt into the fast-access storage, and upon repeating the query it will already be available for download.

8.3.2 Technical description of the service

The receipt query provided by the receipt repository is implemented through the "getDocument" service call.

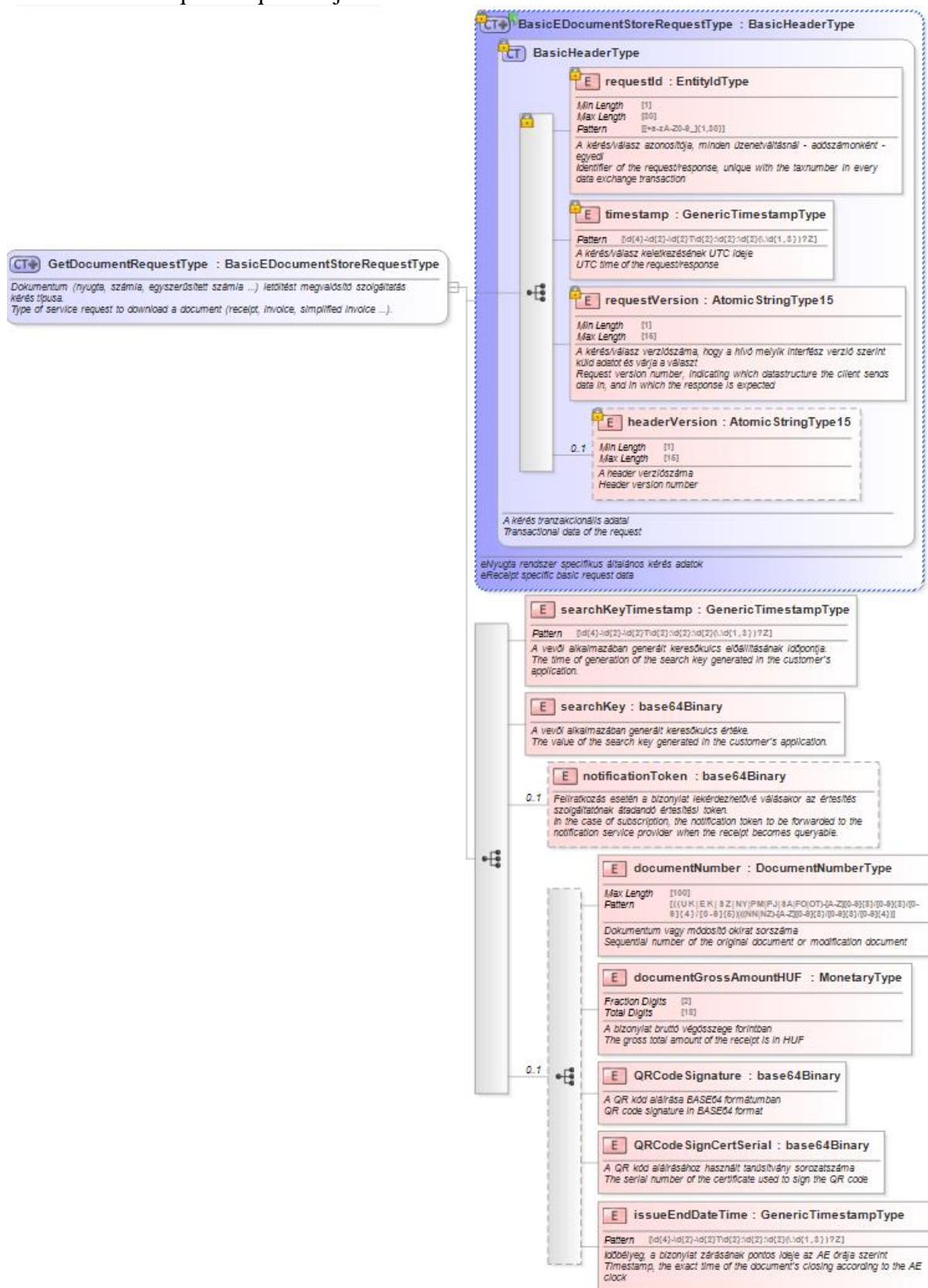
- Context root: /eDocStore/v1
- URL: /getDocument



- Request object: GetDocumentRequest. The technological description of the service's request object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.
- Response object: GetDocumentResponse. The technological description of the service's response object can be found in the "[Description of business data content \(XSD model types and elements\)](#)" section.

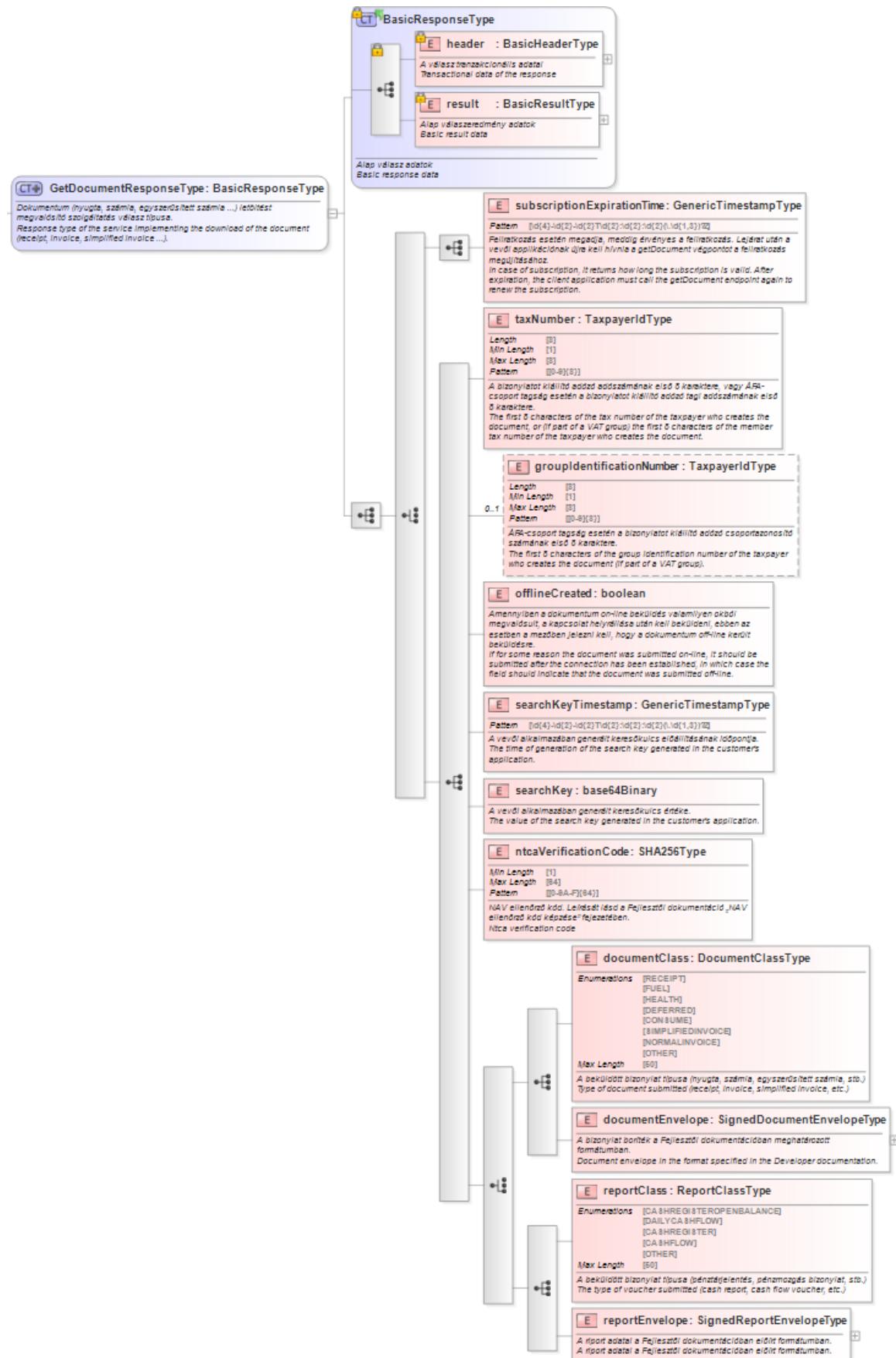


GetDocumentRequest request object:





GetDocumentResponseType object:





8.4 Submission of events occurring in the customer application

During its operation, the customer application automatically and anonymously reports unexpected events encountered during receipt download or processing to the receipt repository. This greatly aids in handling erroneous cases and eliminates the need for user reports.

8.4.1 Business description of the service

The purpose of the service is to collect cases that deviate from normal operation for further investigation, thereby continuously improving the quality of e-cash registers and customer applications.

The customer application may only submit events related to downloads if the encryption key pair required for receipt encryption was either generated by the e-cash register or generated by the customer application, and the customer application has read and stored the e-cash register's confirmation for the given key.

For encryption key pairs generated by the customer application, there is no guarantee that a receipt must be associated with them, so in this case, a download error must not be reported.

However, errors encountered during the processing of downloaded receipts must be reported regardless of where the key pair was generated.

The customer application may only submit a report to the receipt repository for a given event once.

Data to be submitted:

- Search key
- Event code
- Optional text description
- Receipt serial number
- Receipt issuance timestamp

The event codes related to issues encountered *during receipt download* are as follows:

Event description	Event code
The receipt did not arrive, and all retry download attempts were unsuccessful.	C0001



The event codes related to issues encountered *during receipt processing*:

Event description	Event code
The customer application was unable to download the e-cash register's signing key for verifying the receipt envelope signature.	C0101
The receipt envelope signature verification failed.	C0102
Decryption of the customer envelope failed (thus, the symmetric encryption key and additional data intended for the customer cannot be retrieved).	C0103
The customer envelope contains a warranty reference to an attachment, but no attachment is present in the envelope.	C0104
The customer attachment is not interpretable (cannot be displayed).	C0105
The encryption of the receipt envelope could not be decrypted.	C0106
The decrypted receipt envelope could not be extracted.	C0107
Decompression of the receipt data failed.	C0108
QR code interpretation error. It must be sent if the data content of the QR code does not comply with the e-cash register output QR code specification. The scanned data should only be considered an e-cash register output QR code if the beginning of the content starts with the character sequence "2 ". If a field contains interpretable data, it must be included in the corresponding XML data structure, and for data required by the XSD but not readable from the QR code, a format-compliant but clearly invalid value must be submitted.	C0109

The optional text field should include additional details related to the given event code, e.g., "missing warranty certificate", "coupon expiration date is invalid".

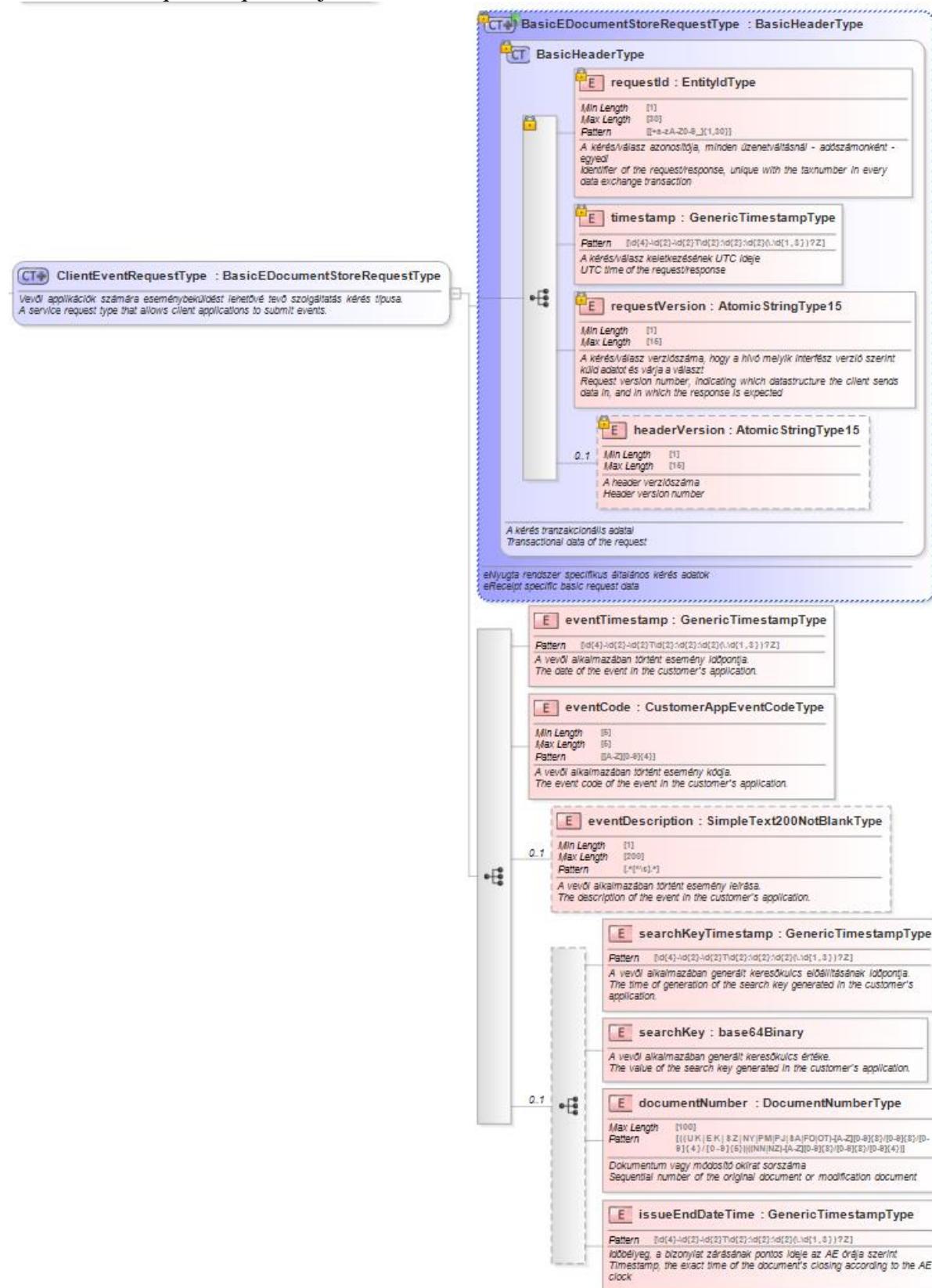
8.4.2 Technical description of the service

The event reception provided by the receipt repository is implemented by calling the "clientEvent" endpoint.

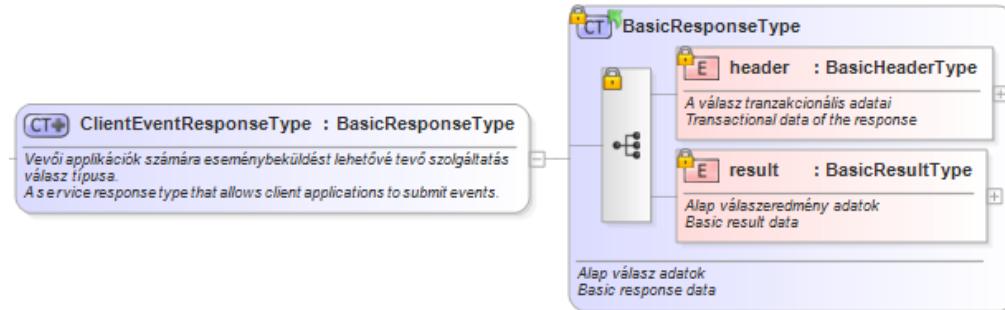
- Context root: /eDocStore/v1
- URL: /clientEvent
- The technological description of the service's request object can be found in the section "[Description of business data content \(XSD model types and elements\)](#)".
- Response object: ClientEventResponse. The technological description of the service's response object can be found in the section "[Description of business data content \(XSD model types and elements\)](#)"



ClientEventRequest request object:



The ClientEventResponseType response object does not return any endpoint-specific data apart from the header and result data.



8.5 Retrieving notification data

The receipt store supports sending push notifications for client applications. When the client application requests the receipt from the receipt store, and the receipt is not yet available for download, the receipt store records the client request ID (“subscription”) for the given receipt. At this endpoint, the client application backend can query which receipts have become available for download since the subscription was recorded.

8.5.1 Business description of the service

The backend can request its related notifications in batch mode by calling the endpoint. The endpoint always returns the notifications received since the last successful request. The frequency of calls is determined by the backend, but it may not be called more often than every two seconds.

The endpoint returns a list of customerId-requestId pairs. The receipt store reads the customerId from the client token (which was submitted by the backend at the application registration endpoint). The customer backend sends the requestId as a push message to the user’s device, therefore it is especially important that the client application stores the identifier used for the request, as this is how it can identify which receipt has become available for download.

If the customer backend can only process a limited number of notifications in one call, it has the option to request the data paginated by specifying the number of rows per page to be returned. The receipt store returns the total number of notifications in the response, so the backend knows that it must request additional pages. The additional pages can be downloaded by repeatedly calling the endpoint with the session identifier provided by the backend in the first call.

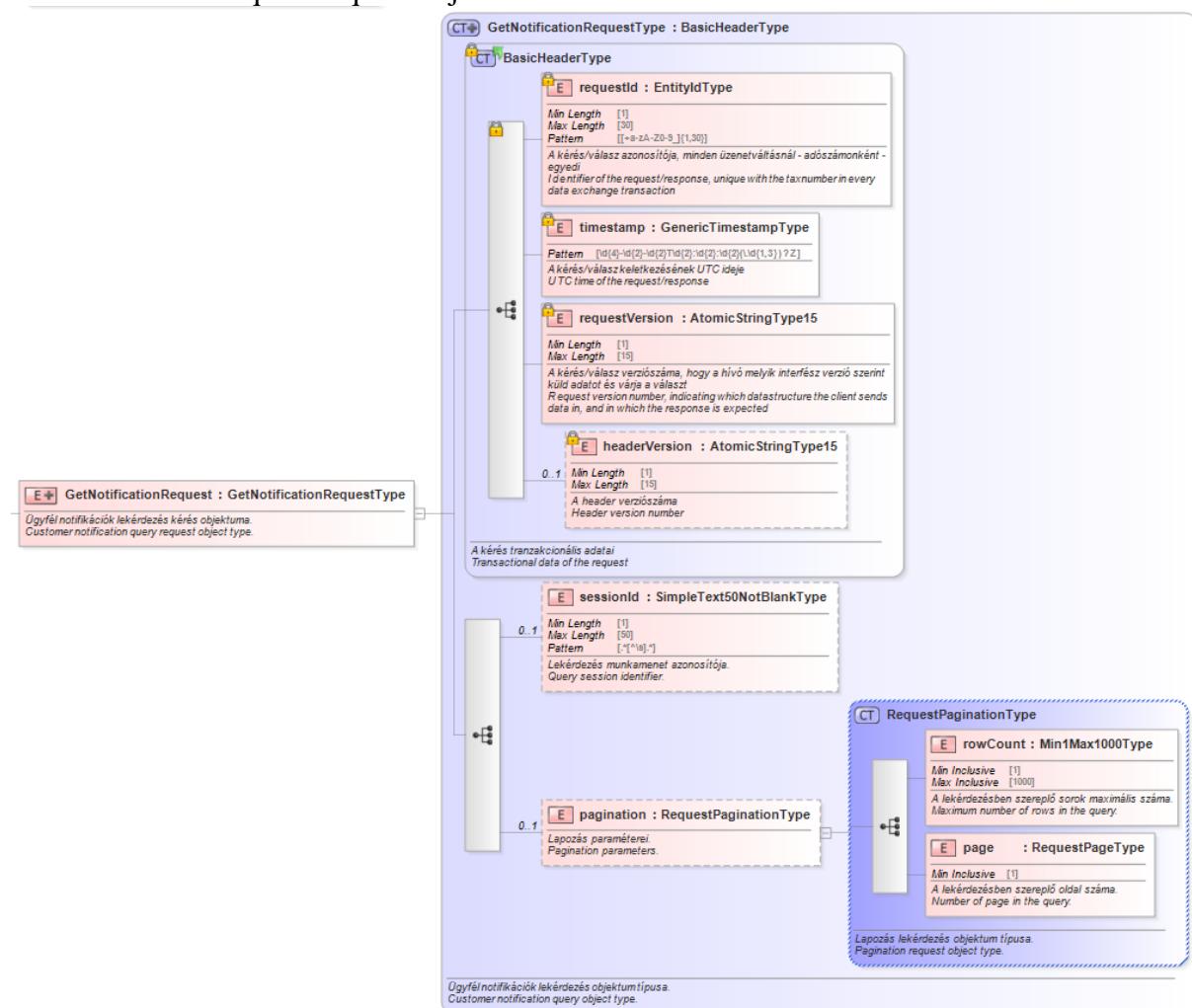
8.5.2 Technical description of the service

The notification download provided by the receipt store is implemented by calling the “getNotification” endpoint.

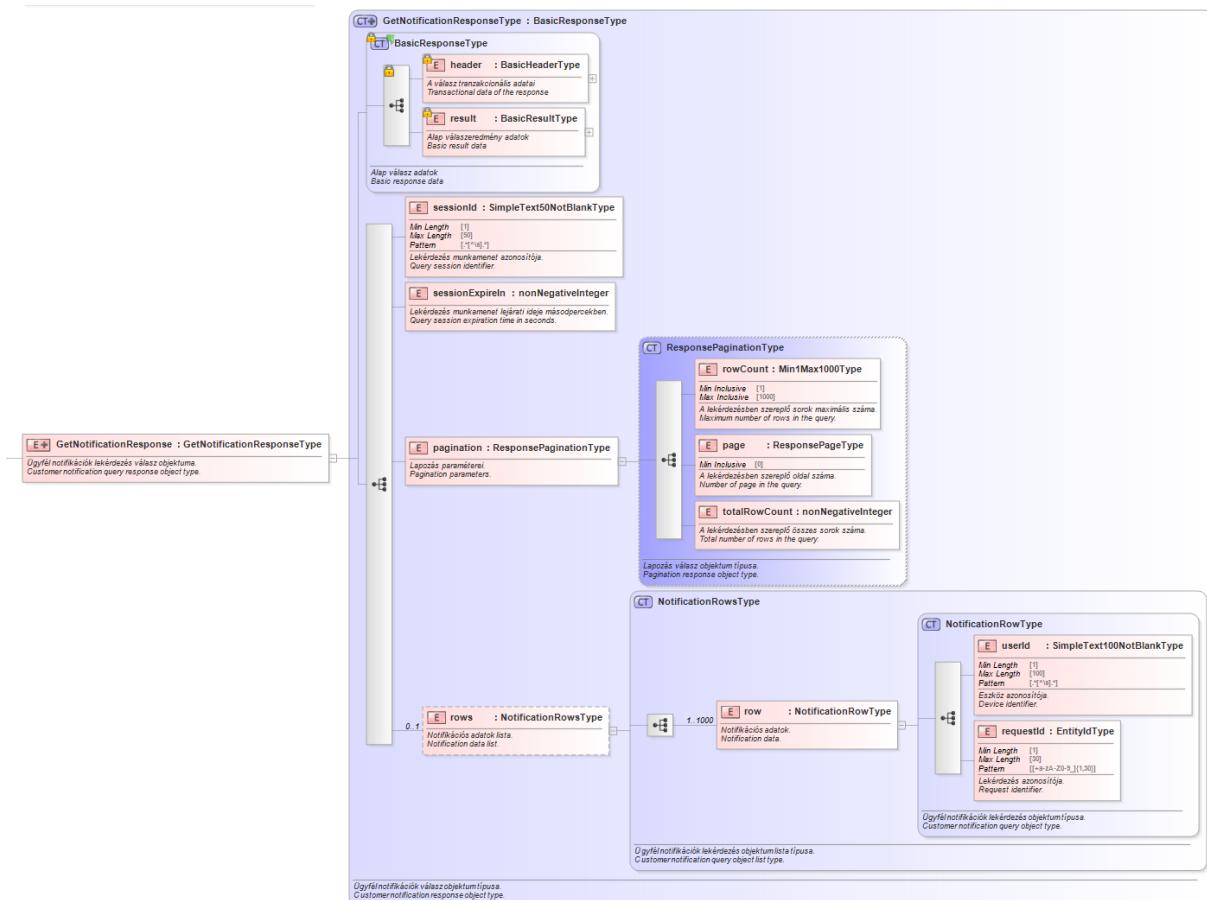
- Context root: /eDocStoreNotif/v1
- URL: /getNotification
- Request object: GetNotificationRequest. The technological description of the service request object can be found in the chapter "[Description of business data content \(XSD model types and elements\)](#)"
- Response object: GetNotificationResponse. The technological description of the service response object can be found in the chapter "[Description of business data content \(XSD model types and elements\)](#)"



GetNotificationRequest request object:



GetNotificationResponse response object:



8.6 Retrieving notification download workflows

At this endpoint, the customer backend can query the active workflows. It is useful if the customer application backend needs to recover after some error, as this way it can find out which query workflows were interrupted and remained in progress.

8.6.1 Business description of the service

The endpoint call has no specific input parameter. In the response message, the receipt store returns the identifiers (sessionId) of the active workflows and their expiration time.

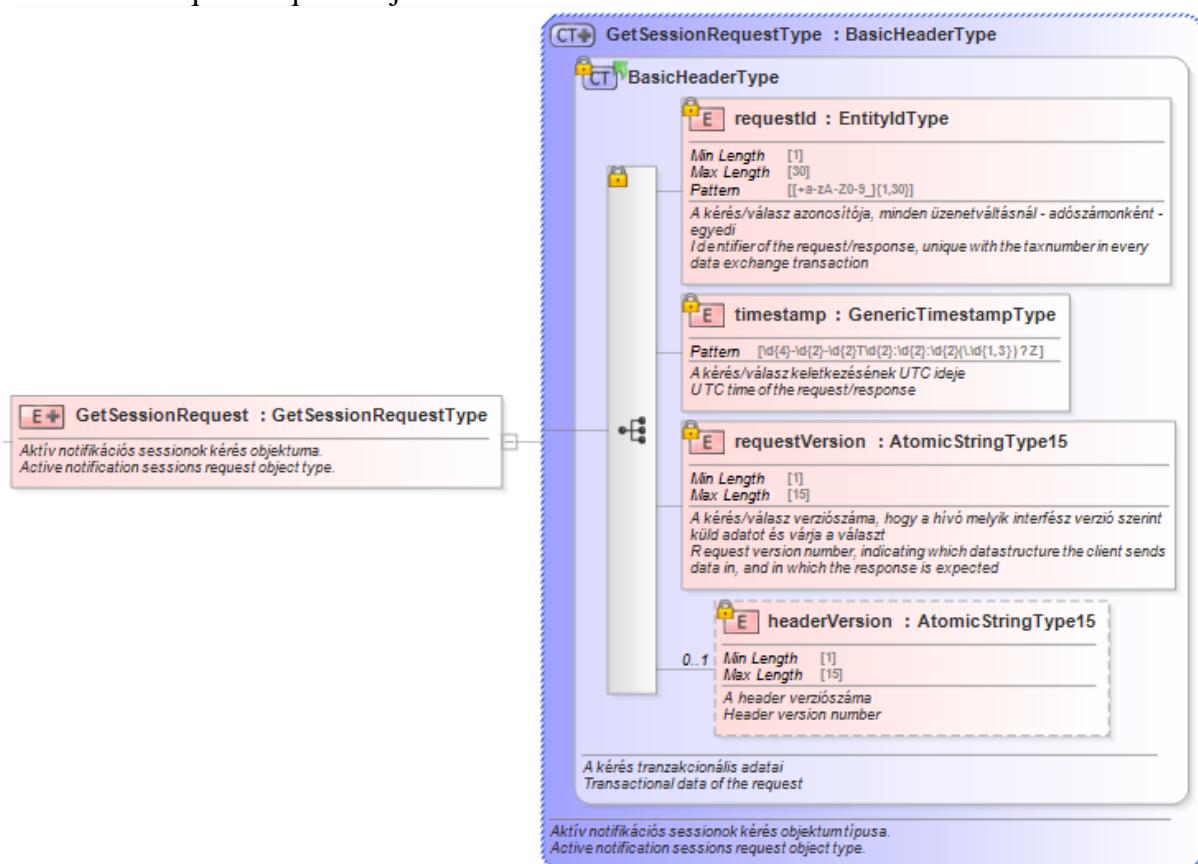
8.6.2 Technical description of the service

The repeated notification download provided by the receipt store is implemented by calling the “getSession” endpoint.

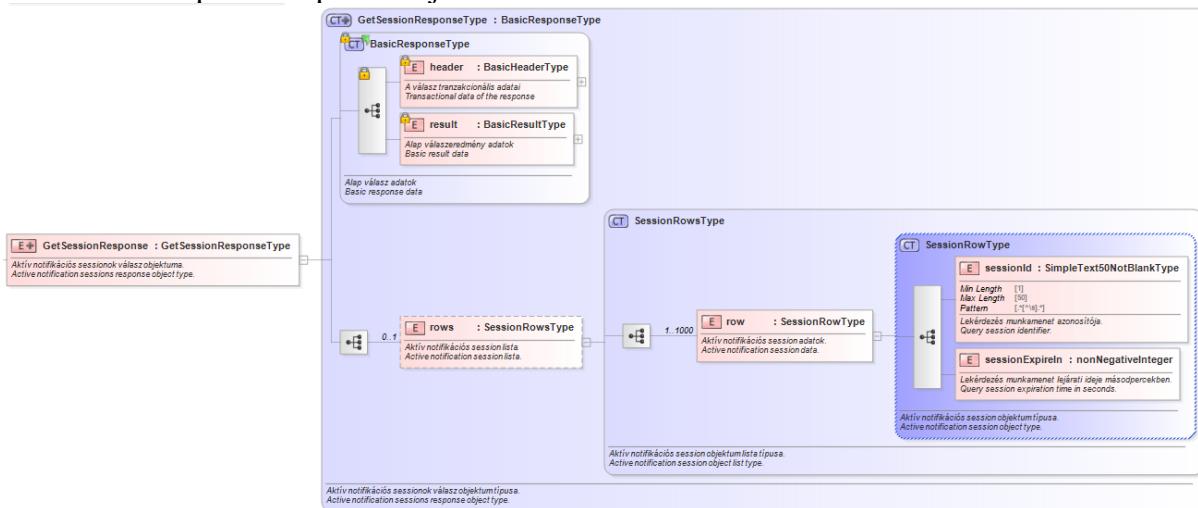
- Context root: /eDocStoreNotif/v1
- URL: /getSession
- Request object: GetSessionRequest. The technological description of the service request object can be found in the chapter "[Description of business data content \(XSD model types and elements\)](#)".
- Response object: GetSessionResponse. The technological description of the service response object can be found in the chapter "[Description of business data content \(XSD model types and elements\)](#)".



GetSessionRequest request object:



GetSessionResponse response object:





9 Storage of receipts on hardware-based E-Cash Registers

The hardware-based e-cash register must store all messages sent to NAV-I and all responses received. The stored data must be an exact match to the sent and received messages in both content and format.

The e-cash register must store the data in a separate file daily, labeled with the tax number and date. Multiple files may be created within a single day; in this case, the files must also be numbered sequentially within the day.

On dual-taxpayer e-cash registers, the data for each taxpayer must be stored in separate files.

Each request must be recorded in chronological order within the file.



10 Description of business data content (XSD model types and elements)

This chapter lists the elements and simple types included in the XSD. The HTML view containing the full visualization of the XSDs is published by NAV on its GitHub page.

10.1 eReceiptApi.xsd

10.1.1 XSD Element list

10.1.1.1 BlockUnblockRequest

The request object for the service that implements the blocking or unblocking of the e-cash register.

10.1.1.2 BlockUnblockResponse

The response object for the service that implements the blocking or unblocking of the e-cash register.

10.1.1.3 CashRegisterInfoRequest

The request type of e-cash register status submission.

10.1.1.4 CashRegisterInfoResponse

The response type for e-cash register status submission

10.1.1.5 CommMgrRequest

The request object for the service that returns the list of services to be invoked by the e-cash register.

10.1.1.6 CommMgrResponse

The response object for the service that returns the list of services to be invoked by the e-cash register.

10.1.1.7 DocumentRequest

The request object for the service that implements the receipt, invoice, simplified invoice, and certain special sales receipt processing.

10.1.1.8 DocumentResponse

The response object for the service that implements the processing of receipts, invoices, simplified invoices, and certain special sales receipts.

10.1.1.9 EndOfOperationRequest

The request object for the service that finalizes the operation of the e-cash register.

10.1.1.10 EndOfOperationResponse

The response object for the service that finalizes the operation of the e-cash register.

10.1.1.11 GetProductByCodeRequest

The request object of the e-cash register operation termination service.

10.1.1.12 GetProductByCodeResponse

The response object of the e-cash register operation termination service.



10.1.1.13 HelloRequest

The request object for the service to be invoked following the registration or re-personalization of the e-cash register. The service call notifies NAV-I that the registration or re-personalization process has been completed.

10.1.1.14 HelloResponse

The response object for the service to be invoked following the registration or re-personalization of the e-cash register.

10.1.1.15 OperatorSiteUpdateRequest

The request object for the service that updates the operator or operating site data on the e-cash register.

10.1.1.16 OperatorSiteUpdateResponse

The response object for the service that updates the operator or operating site data on the e-cash register.

10.1.1.17 OwnerChangeRequest

The request object for the service that initiates the re-personalization of the e-cash register.

10.1.1.18 OwnerChangeResponse

The response object for the service that initiates the re-personalization of the e-cash register.

10.1.1.19 PrintTechnicalInfoRequest

The request object for the service that sends a technical information receipt.

10.1.1.20 PrintTechnicalInfoResponse

The response object for the service that sends a technical information receipt.

10.1.1.21 QueryCertificateResponse

The response type for querying the issued certificate.

10.1.1.22 QueryTaxpayerRequest

Request object of the domestic tax number verification query service .

10.1.1.23 QueryTaxpayerResponse

Response object of the domestic tax number query service.

10.1.1.24 RegistrationRequest

The request object for the service that implements device registration.

10.1.1.25 RegistrationResponse

The response object for the service that implements device registration.

10.1.1.26 RenewCertificateRequest

The request object for the service that initiates the renewal of an authentication certificate.

10.1.1.27 RenewCertificateResponse

The response object for the service that initiates the renewal of an authentication certificate.



10.1.1.28ReportRequest

The request object for the service that processes the opening receipt, daily transaction report, cash register report, cash movement receipt, and all other non-tax-related receipts.

10.1.1.29ReportResponse

The response object for the service that processes the opening receipt, daily transaction report, cash register report, cash movement receipt, and all other non-tax-related receipts.

10.1.1.30SoftwareUpdateRequest

The request object for the service that initiates the update of the software running on the e-cash register.

10.1.1.31SoftwareUpdateResponse

The response object for the service that initiates the update of the software running on the e-cash register.

10.1.1.32VatUpdateRequest

The request object for the service that modifies the VAT rates associated with specific transaction categories.

10.1.1.33VatUpdateResponse

The response object for the service that modifies the VAT rates associated with specific transaction categories.

10.1.1.34SendMissingDocumentRequest

The request object for the service that submits a missing document.

10.1.1.35SendMissingDocumentResponse

The response object for the service that submits a missing document.

10.2 documentMessage.xsd

10.2.1 XSD Element list

10.2.1.1 CustomerDocument

The receipt and receipt attachment data to be handed over to the customer.

10.2.1.2 CoreDocument

The receipt and control data to be submitted to NAV

10.3 reportMessage.xsd

10.3.1 XSD Element lista

10.3.1.1 CoreReport

The report to be submitted to NAV.

10.3.1.2 CustomerReport

The report attachment data to be handed over to the customer.



10.4 documentData.xsd

10.4.1 XSD Simple type lista

10.4.1.1 LineNatureIndicatorType

common:AtomicStringType15

The type indicating whether a given line item pertains to product sales or service provision

Restriction code	Value
required	No

Enum code	Enum description
PRODUCT	Product sales
SERVICE	Service provision
OTHER	Other, not classifiable

10.4.1.2 ProductCodeCategoryType

common:AtomicStringType8

Type for indicating the category of the product code

Restriction code	Value
Megszorítás kód	Value
minLength	2
required	No
maxLength	6

Enum code	Enum description
EAN	EAN code (European Article Number)
VTSZ	Customs tariff number (Vámtarifa szám - VTSZ)
SZJ	Service classification number (Szolgáltatás jegyzék szám - SZJ)
KN	KN code (Combined Nomenclature, Annex I of Regulation 2658/87/EEC)
AHK	Administrative reference code (AHK) under the Excise Tax Law (Act LXVIII of 2016)
CSK	Packaging material catalog code (CsK code) according to Annex 1, Section A of Government Decree 343/2011 (XII. 29.)
KT	Environmental product code (Kt code) according to Annex 1, Section B of Government Decree 343/2011 (XII. 29.)
EJ	Construction register number (EJ)
TESZOR	Product code according to the Classification System of Products and Services (TESZOR) – Regulation 451/2008/EC
OWN	Product code created by the company
OTHER	Other product code



10.4.1.3 ProductCodeValueType

common:AtomicStringType32

Product code type

Restriction code	Value
minLength	2
required	No
maxLength	30
pattern	[A-Z0-9]{2,30}

10.4.1.4 QuantityType

common:GenericDecimalType

Quantity value type. A maximum of 22 digits, which may include up to 10 decimal places.

Restriction code	Value
totalDigits	22
required	No
fractionDigits	10

10.5 communicationData.xsd

10.5.1 XSD Simple type list

10.5.1.1 UnitOfMeasureType

common:AtomicStringType15

Quantity unit type:

Restriction code	Value
required	No

Enum code	Enum description
PIECE	Piece
KILOGRAM	Kilogram
TON	Metric ton
KWH	Kilowatt hour
DAY	Day
HOUR	Hour
MINUTE	Minute
MONTH	Month
LITER	Liter
KILOMETER	Kilometer
CUBIC_METER	Cubic meter
SQUARE_METER	Square meter
METER	Meter
LINEAR_METER	Linear meter
CARTON	Carton
PACK	Pack
OWN	Custom quantity unit name



10.5.1.2 ProductStateType

common:AtomicString50

Unit of measure type

Restriction code	Value
required	No

Enum code	Enum description
ACTIVE	Active

10.5.1.3 BlockUnblockType

xs:string

Blocking type

Restriction code	Value
maxLength	7
required	No

Enum code	Enum description
BLOCK	Blocking
UNBLOCK	Unblocking

10.5.1.4 CashRegisterWorkStateType

xs:string

e-Cash Register Status Type

Restriction code	Value
maxLength	5
required	No

Enum code	Enum description
OK	The e-cash register is operational and not blocked.
BLOCK	The e-cash register is in a blocked state.
ERROR	The e-cash register is faulty.

10.5.1.5 AeBlockUnblockStateType

xs:string

AE Blocking Status Type

Restriction code	Value
maxLength	7
required	No

Enum code	Enum description
UNBLOCK	Az AE is operational and not blocked
BLOCK	Az AE is in a blocked state.



10.5.1.6 GpsType

xs:string

GPS coordinates according to WGS84 in degrees, minutes, and seconds format.

Restriction code	Value
maxLength	50
required	No
pattern	[0-9]{0,2},[0-9]{4,16} ([0-8][0-9]),([0-5][0-9]),([0-5][0-9])\.(0-9)[0-9])

10.5.1.7 HttpMethodType

xs:string

List of HTTP Methods.

Restriction code	Value
maxLength	4
required	No

Enum code	Enum description
GET	
POST	

10.5.1.8 IncorporationType

common:AtomicStringType15

Economic type

Restriction code	Value
required	No

Enum code	Enum description
ORGANIZATION	Organization
SELF_EMPLOYED	Self employed
TAXABLE_PERSON	Taxable person

10.5.1.9 MobileConnectionType

xs:string

Mobile connection technology type

Restriction code	Value
maxLength	4
required	No

Enum code	Enum description
GSM	GSM
2G	2G technology
2,5G	2,5G technology
3G	3G technology
4G	4G technology
5G	5G technology



10.5.1.10 PercentageType

xs:decimal

Percentage value without the percent sign, using a decimal point if necessary, rounded to a maximum of 2 decimal places

Restriction code	Value
totalDigits	5
required	No
fractionDigits	2
maxInclusive	100

10.5.1.11 PrintMessageType

xs:string

A technical information message transmitted via the interface, which must be printed on the system printer (PRINT) after the cash register closure.

Restriction code	Value
maxLength	2048
required	No
pattern	[a-zA-Z0-9aáééííóóőőúúüü]AÁEÉÍÍOÓÖÖUÚÜÜ_*%=%+§/\@\]{0,2048}

10.5.1.12 CertificateSigningRequestType

base:AtomicCsrType

Certificate Signing Request (CSR) Type in DER format.

10.5.1.13 CMSCertificateSigningRequestType

xs:base64Binary

CMS Certificate Signing Request (CSR) Type in Base64 format.

10.5.1.14 CertificateType

xs:base64Binary

Issued certificate in DER format.

10.5.1.15 CertificateTypeType

xs:string

Certificate type

Restriction code	Value
maxLength	14
required	No

Enum code	Enum description
AUTHENTICATION	Authentication
SIGNING	Signing



10.5.1.16QueryCertificateResultType

xs:string

The result of the certificate query

Restriction code	Value
maxLength	11
required	No

Enum code	Enum description
COMPLETE	Complete
IN_PROGRESS	In progress
ERROR	Error

10.5.1.17RegistrationNumberType

xs:string

Commissioning Code Type

Restriction code	Value
maxLength	16
required	No
pattern	[0-9]{16}

10.5.1.18ServiceType

xs:string

Service list type

Restriction code	Value
maxLength	30
required	No

Enum code	Enum description
BLOCK_UNBLOCK	Cash register blocking/unblocking
DOWNLOAD_PRODUCT_LIST	Product list download
DOWNLOAD_SOFTWARE_UPDATE	Downloading software update
END_OF_OPERATION	End of operation
QUERY_CERTIFICATE	Query certificate
HELLO	Hello service
OPERATOR_SITE_UPDATE	Operator site update
OWNER_CHANGE	Owner change
PRINT_TECHNICAL_INFO	Print technical info
QUERY_TAXPAYER	Domestic Tax Number Verification
REGISTRATION	Device registration
RENEW_CERTIFICATE	Renew certificate (authentication)
RENEW_EXPIRED_CERTIFICATE	Renew expired (authentication) certificate
SOFTWARE_UPDATE	Software update
VAT_UPDATE	VAT Update
SOUND_DOWNLOAD	Downloading sound file
LOGO_DOWNLOAD	Logo download
CASH_REGISTER_INFO	Cash register info download
SEND_MISSING_DOCUMENT	Sending missing document



10.5.1.19 SoftwareIdType

xs:string

Program Identifier Type

Restriction code	Value
length	18
required	No
pattern	[0-9A-Z\-\]{18}

10.5.1.20 SoftwareOperationType

xs:string

The operational type of the e-Cash Register (AE-based (hardware) or online service)

Restriction code	Value
maxLength	16
required	No

Enum code	Enum description
AE	AE-based (hardware) e-cash register
ONLINE_ANDROID	Online Android (cloud-based) e-cash register
ONLINE_IOS	Online iOS (cloud-based) e-cash register
ONLINE_HUAWEI	Online Huawei (cloud-based) e-cash register
ONLINE_SERVICE	Online other (cloud-based) e-cash register

10.5.1.21 TaxpayerAddressClassType

common:AtomicStringType8

Taxpayer address type

Restriction code	Value
required	No

Enum code	Enum description
HQ	HQ
SITE	Site
BRANCH	Branch

10.5.1.22 TeaorCodeType

xs:string

TEÁOR code type

Restriction code	Value
pattern	[0-9]{2} [0-9]{2}.[0-9]{1} [0-9]{2}.[0-9]{2}
required	No
minLength	2
maxLength	5



10.5.1.23 UrlType

xs:string

The standard URL of the service to be invoked by the e-cash register (in the format https://host:port/resource URI).

Restriction code	Value
maxLength	2000
required	No

10.5.1.24 FiscalDayStateType

xs:string

Fiscal day status type

Restriction code	Value
maxLength	6
required	No

Enum code	Enum description
OPENED	Opened fiscal day
CLOSED	Closed fiscal day

10.5.1.25 EventCodeType

xs:string

Event code type

Restriction code	Value
maxLength	30
required	No

Enum code	Enum description
POWER_ON	Cash register power on
SHUTDOWN	Cash register shutdown
BLOCK	Cash register blocked itself due to an error
UNBLOCK	End of blocked state due to an error
MESSAGE_ACK	Acknowledgment of technical message sent by NAV for display/printing
OTHER_EVENT	Reporting of other important event

10.6 eReceiptBase.xsd

10.6.1 XSD Simple type list

10.6.1.1 APNumberType

xs:string

AP number type

Restriction code	Value
minLength	9
required	No
maxLength	9
pattern	[A-Z][0-9]{8}



10.6.1.2 CustomerAppEventType

xs:string

Customer application event code type

Restriction code	Value
minLength	5
required	No
maxLength	5
pattern	[A-Z][0-9]{4}

10.6.1.3 Digit4Type

xs:string

4 digit number type

Restriction code	Value
minLength	4
required	No
maxLength	4
pattern	[0-9]{4}

10.6.1.4 EANType

xs:string

8 or 13 digit EAN code type

Restriction code	Value
minLength	8
required	No
maxLength	13
pattern	[0-9]{8} [0-9]{13}

10.6.1.5 ProcessIdentifierType

xs:string

EPD process identifier

Restriction code	Value
maxLength	10
required	No
minLength	10
pattern	[A-Z0-9]{10}

10.6.1.6 AtomicCsrType

xs:base64Binary

Atomic CSR type

Restriction code	Value
minLength	1
required	No
maxLength	8192
pattern	.*[^\s].*



10.6.1.7 File512kBinaryType

xs:base64Binary

Binary file type, max 512kB

Restriction code	Value
maxLength	524288
required	No

10.6.1.8 FileExtensionType

xs:string

Fájl extension type

Restriction code	Value
minLength	1
required	No
maxLength	10
pattern	[\\w,-]*

10.6.1.9 CancellationReasonType

common:AtomicStringType15

Cancellation reason type

Restriction code	Value
required	No

Enum code	Enum description
S1	Customer withdrawal
S2	Operator error: incorrect input
S3	Operator error: incorrect payment method entry
S4	Operator error: product out of stock
S5	Technical: incorrect receipt type issued
S6	Technical: unsuccessful payment method usage
S7	Technical: incorrect customer data/invalid input
S8	Technical: test purchase
S0	Other

10.6.1.10 CashPaymentTitleType

common:AtomicStringType15

Cash Register deposit-withdrawal or payment method exchange reason type

Restriction code	Value
required	No
Enum code	Enum description
01	Change money input
02	Cashier cash withdrawal
03	Fee collection



Enum code	Enum description
04	Lottery ticket sales
05	Advance payment
06	Cash register shortage
07	Tip
08	Other deposit
31	Skimming (cash removal)
32	Cashier removal
33	Voucher withdrawal
34	Gift card withdrawal
35	Salary payment
36	Salary advance
37	Postage cost
38	Other overhead costs
39	Goods purchase
40	Closing balance removal
41	Other payment
42	Cash withdrawal
60	Payment method exchange

10.6.1.11 CollectorCodeType

xs:string

Collector code type (revenue collection type)

Restriction code	Value
maxLength	3
required	No
pattern	[A-E]N TAM AAM EAM ATK TRA SEC ART ANT EUE HO

10.6.1.12 CustomerVatStatusType

common:AtomicString Type15

Customer VAT status type

Restriction code	Value
required	No

Enum code	Enum description
DOMESTIC	Domestic VAT entity
OTHER	Other (domestic non-VAT entity, non-natural person, foreign VAT entity, and foreign non-VAT entity, non-natural person)
PRIVATE_PERSON	Non-VAT entity (domestic or foreign) natural person

10.6.1.13 DataNameType

common:AtomicString Type255

Unique Identifier Type for Data Field



Restriction code	Value
minLength	1
required	No
maxLength	255
pattern	[A-Z][0-9]{5}[_][_A-Z0-9]{1,249}

10.6.1.14 DocumentCategoryType

xs:string

Document category type

Restriction code	Value
maxLength	50
required	No

Enum code	Enum description
RECEIPT	Receipt
HEALTH	Health
FUEL	Fuel
CONSUME	Consume
OTHER	Other
DEFERRED	Deferred

10.6.1.15 DocumentClassType

xs:string

Receipt type, specifying which receipt the e-cash register submits

Restriction code	Value
maxLength	50
required	No

Enum code	Enum description
RECEIPT	Receipt document
FUEL	Fuel card sales receipt
HEALTH	Health card sales receipt
DEFERRED	Hotel charge transfer receipt
CONSUME	Consumption summary receipt
SIMPLIFIEDINVOICE	Simplified invoice
NORMALINVOICE	Normal invoice
OTHER	Other receipt

10.6.1.16 DocumentDateType

xs:date

Document date type

Restriction code	Value
minInclusive	2010-01-01
required	No
pattern	\d{4}-\d{2}-\d{2}



10.6.1.17 DocumentNumberType

xs:string

Document or amendment deed serial number type

Restriction code	Value
maxLength	100
required	No
pattern	((UK EK SZ NY PM PJ HC FB QQ)-[A-Z][0-9]{8}/[0-9]{8}/[0-9]{4}/[0-9]{5}) ((NN NZ)-[A-Z][0-9]{8}/[0-9]{8}/[0-9]{4})

10.6.1.18 DocumentOperationType

common:AtomicStringType8

Document operation type

Restriction code	Value
required	No

Enum code	Enum description
CREATE	Original document
MODIFY	Amendment Deed to the Original Document
STORNO	Cancellation of the Original Document

10.6.1.19 DocumentUnboundedIndexType

xs:int

Serial number type

Restriction code	Value
minInclusive	1
required	No

10.6.1.20 EncryptedSymmetricInitialVectorType

xs:base64Binary

Symmetric key type used for encrypting the supplementary part of the receipt.

10.6.1.21 EncryptedSymmetricKeyType

xs:base64Binary

Symmetric key type used for encrypting the core part of the receipt

10.6.1.22 ItemNatureType

xs:string

Item nature type

Restriction code	Value
maxLength	5
required	No

Enum code	Enum description
n	Sale
ns	Sale cancellation (storno)
e	Discount



Enum code	Enum description
es	Discount cancellation (storno)
k	Non-business policy discount
ks	Non-business policy discount cancellation (storno)
f	Surcharge
fs	Surcharge cancellation (storno)
g	Returnable packaging refund
gs	Returnable packaging refund cancellation (storno)
v	Returned goods
vs	Returned goods cancellation (storno)
x	Consumption summary
xs	Consumption summary cancellation (storno)
p	Cash movement receipt (for customer-related transactions)

10.6.1.23 LineNumberType

xs:nonNegativeInteger

Line number type

Restriction code	Value
minInclusive	1
required	No
totalDigits	20

10.6.1.24 ModificationReasonType

common:AtomicStringType15

Return reason type

Restriction code	Value
required	No

Enum code	Enum description
V1	Defective goods
V2	Customer withdrawal from purchase
V3	Other

10.6.1.25 MonetaryType

common:GenericDecimalType

Monetary value type. Maximum 18 digits, with up to 2 decimal places.

Restriction code	Value
totalDigits	18
required	No
fractionDigits	2



10.6.1.26 ExchangeRateType

xs:decimal

Exchange rate type

Restriction code	Value
totalDigits	14
required	No
fractionDigits	6
minExclusive	0

10.6.1.27 NtcaControlCodeType

xs:string

NAV verification code type on the receipt

Restriction code	Value
maxLength	5
required	No
pattern	[0-9A-F]{5}

10.6.1.28 PaymentMethodType

common:AtomicStringType15

Payment method type

Restriction code	Value
required	No

Enum code	Enum description
SZEP	SZÉP card
CASH	Cash
CARD	Bank card, credit card, other cash substitute instrument.
AFR	Instant Payment System
OTHER	Other payment type

10.6.1.29 ReportClassType

xs:string

Report Receipt Type (which report the e-cash register submits).

Restriction code	Value
maxLength	50
required	No

Enum code	Enum description
CASHREGISTEROPENBALANCE	Cash Register Opening Receipt
DAILYCASHFLOW	Daily Sales Report
CASHREGISTER	Cash Register Report
CASHFLOW	Cash Movement Receipt
OTHER	Other



10.7 eDocumentStoreApi.xsd

10.7.1 XSD Element list

10.7.1.1 AppRegistrationRequest

The request object for the service that implements customer application registration.

10.7.1.2 AppRegistrationResponse

The response object for the service that implements customer application registration.

10.7.1.3 GetDocumentRequest

The request object for the service that enables document (receipt, invoice, simplified invoice, etc.) download.

10.7.1.4 GetDocumentResponse

The response object for the service that enables document (receipt, invoice, simplified invoice, etc.) download.

10.7.1.5 ClientEventRequest

The request object for the service that allows event submission from customer applications.

10.7.1.6 ClientEventResponse

The response object for the service that allows event submission from customer applications.

10.8 eDocumentStoreMessage.xsd

10.8.1 XSD Simple type list

10.8.1.1 AtomicStringType8192

xs:string

Atomic string type for 8192 lenght

Restriction code	Value
minLength	1
required	No
maxLength	8192

10.8.1.2 JwtTokenType

AtomicStringType8192

Restriction code	Value
pattern	.*[^\s].*
required	No

10.9 eReceiptExport.xsd

10.9.1 XSD Element list

10.9.1.1 ExportEnvelope

Export envelope.



11 Error handling

The service operates with a shared set of enumerated values on the service side, including a list of result and error codes.

Unlike result codes, error codes are intentionally not included in the schema descriptor enumerations to prevent implementation dependencies on the client side in case of changes or expansions. Result codes are returned in the funcCode tag of the BasicResultType node, and Error codes are returned in the errorCode tag of the response message.

The received funcCode values should be interpreted according to the corresponding business process being executed.

11.1 Error response

In the case of an error response, the service returns the response object corresponding to the given service; however, the response object does not contain the fields carrying the business result, only the BasicEReceiptResponseType element is returned. Within the BasicEReceiptResponseType element, the BasicResultType element contains the error-related data. The funcCode always contains the value ERROR in the case of an error response.

11.2 Technical error codes

Upon request reception, the entire request is verified, except for the "[Receipt Submission](#)" and "[Report reception](#)" interfaces. In the case of "[Receipt Submission](#)" and "[Report reception](#)" the BASE64-encoded business content within the envelopes is not verified at this step.

Errors are categorized based on their nature as "transient" (temporary) or "permanent" errors.

A transient error occurs due to the state of the server rather than the submitted data content. In such cases, the request may be resubmitted with the same data content, updating the request timestamp, after waiting for the specified retry period associated with the error code.

A permanent error relates to the data content, meaning that a field contains a value that violates content-related rules, making its acceptance impossible. In this case, the request cannot be resubmitted with the same data content, and the indicated field value must be reviewed and corrected.

11.2.1 Common error codes

The following table summarizes the error codes applicable to both the e-cash register and NAV interface, as well as the receipt repository interface.

HTTP Response	Error Code	Message	Error Reason	Type	Nature	Explanation / Action
HTTP 404 Not Found	N/A	N/A	Incorrect service endpoint in the request	K	P	Check the URL used for the service call.
HTTP 400 Bad Request	N/A	N/A	Malformed XML in the request message	K	P	A syntactically incorrect XML message must not be considered or processed as



HTTP Response	Error Code	Message	Error Reason	Type	Nature	Explanation / Action
						XML according to the XML standard; it must be corrected.
HTTP 422 Unprocessable Content	M0002	Invalid request	Non-schema-valid XML in the request body	K	P	The submitted XML contains elements that violate the XSD constraints (as listed in the response); they must be corrected.
HTTP 503 Service Unavailable	M0001	Maintenance mode	Maintenance is in progress	Sz	T	The requested operation is temporarily unavailable due to maintenance. Follow the announcements on the platform and retry the request later. At least 30 minutes must pass between the original and repeated requests.
HTTP 422 Unprocessable Content	M0003	Invalid timestamp	The timestamp in the request is too old or in the future	K	P	The timestamp in the request falls outside the allowed tolerance, which is currently ± 1 hour from the server time. The request can be retried immediately after updating the client's system clock.
HTTP 500 Internal Server Error	N/A	Operation failed	Unexpected processing error	Sz	T	The request must be retried later. At least 30 minutes must pass between the original and repeated requests. If the error persists, report it to the NAV helpdesk, but first, check the portal for any announcements about system outages or maintenance.
HTTP 422 Unprocessable Content	B0002	Invalid request version	The request version number is incorrect	K	P	Only version "1.0" is currently accepted.
HTTP 422 Unprocessable Content	B0003	Request ID not unique	The requestId in the request is not unique	K	P	The provided requestId has already been used for the given AP number. A new, unique requestId must be specified.



HTTP Response	Error Code	Message	Error Reason	Type	Nature	Explanation / Action
HTTP 422 Unprocessable Content	M0009	Invalid search key timestamp	The searchKeyTimestamp in the request is too old or in the future	K	P	The searchKeyTimestamp value in the request falls outside the allowed range. The searchKeyTimestamp cannot be older than January 1, 2024, and it cannot be later than +1 hour from the server time.

11.2.2 ECash register interface error codes

Response codes for Between the E-Cash register and NAV.

HTTP Response	Error Code	Message	Error Reason	Type	Nature	Explanation / Action
HTTP 403 Forbidden	B0001	IP address error	IP address issue	Sz	T	The request must be retried at a later time. At least 30 minutes must pass between the original and repeated requests.
HTTP 403 Forbidden	M0004	AP number conflict auth cert	The AP number in the request does not match the one in the authentication certificate	K	P	The request message must contain the same AP number as in the client certificate.
HTTP 403 Forbidden	M0005	Tax number conflict auth cert	The tax number in the request does not match the one in the authentication certificate	K	P	The request message must contain the same tax number as in the client certificate.
HTTP 403 Forbidden	M0007	AP number conflict signing cert	The AP number in the request does not match the one in the signing certificate	K	P	The request message must contain the same AP number as in the signing certificate used for signing the submitted receipt envelope.
HTTP 403 Forbidden	M0008	Tax number conflict signing cert	The tax number in the request does not match the one in the signing certificate	K	P	The request message must contain the same tax number as in the signing certificate used for signing the submitted receipt envelope.
HTTP 422 Unprocessable Content	B0004	Record counter not unique	The recordCounter in the request is not unique	K	P	The recordCounter provided for the given AP number and taxpayer has already been submitted, and no process identifier for repeated submission has been specified. Either a new recordCounter must be



HTTP Response	Error Code	Message	Error Reason	Type	Nature	Explanation / Action
						provided, or a process identifier must be included.
HTTP 422 Unprocessable Content	B0005	Invalid last record counter	The lastRecordCounter in the request is incorrect	K	P	The lastRecordCounter in the request cannot be smaller than the recordCounter and must not be smaller than the previously submitted lastRecordCounter for the given AP number and taxpayer.
HTTP 422 Unprocessable Content	N/A	End-of-operation not permitted	Termination of operation is not allowed	Sz	T	The given e-cash register does not meet the conditions required for the "End of Operation" service; therefore, termination of operation cannot be executed.
HTTP 422 Unprocessable Content	B2001	Invalid license number	Invalid license number in AP number	K	P	The e-cash register is attempting to register with an AP number containing a non-existent license number. The process can be restarted with a valid AP number containing a correct license number.
HTTP 422 Unprocessable Content	B2002	Taxpayer mode mismatch	Inconsistent dual-business mode	K	P	The commissioning code provided during device registration does not match the registered business number in the license number, meaning a single-business e-cash register was attempted to be registered with a dual-business commissioning code, or vice versa. The process must be restarted with a commissioning code matching the business number in the license number.
HTTP 422 Unprocessable Content	B2003	Invalid AP number	The cash register is already registered	K	P	A completed device registration has already been performed with the given AP number. The process must be restarted with a new, non-existent AP number.



HTTP Response	Error Code	Message	Error Reason	Type	Nature	Explanation / Action
HTTP 422 Unprocessable Content	B2004	Restarted registration data mismatch	Restarting registration with different data is not allowed	K	P	Restarting an incomplete device registration is only allowed with the same IMEI and IMSI pair, AP number, and commissioning code. The process must be restarted using the data provided during the first registration attempt.
HTTP 422 Unprocessable Content	B2005	Invalid registration code	The commissioning code (ÜH code) is no longer usable	K	P	The commissioning code does not exist or has already been used. The process can be restarted with a valid, unused commissioning code.
HTTP 422 Unprocessable Content	B2006	IMSI-IMEI mismatch	Invalid IMSI and IMEI pairing	K	P	The provided IMSI and IMEI values were not assigned to an AE during the manufacturer's device registration in the NAV system. The AE device registration file must be verified.
HTTP 422 Unprocessable Content	B2011	Invalid software version	Unregistered software (hash)	K	P	Device registration can only be performed using software authorized for the given license number. The process can be restarted after installing the authorized software.
HTTP 422 Unprocessable Content	B2012	Misused hello during registration	Commissioning cannot be finalized (new device registration)	K	P	The hello endpoint can only be called after the device registration endpoint call and successful certificate download. The necessary steps must be completed before retrying the hello call.
HTTP 422 Unprocessable Content	B2013	Misused hello during owner change	Commissioning cannot be finalized (re-personalization)	K	P	During re-personalization, the hello endpoint can only be called after successfully completing the "End of Operation" process. The necessary steps must be completed before retrying the hello call.
HTTP 422 Unprocessable Content	B2023	Unsent document or report	Missing receipt/report	K	T	The e-cash register has not submitted all required receipts to NAV, preventing the completion of the



HTTP Response	Error Code	Message	Error Reason	Type	Nature	Explanation / Action
HTTP 422 Unprocessable Content						operation. The process can be restarted after submitting all necessary receipts and reports.
HTTP 422 Unprocessable Content	B2024	Owner change not permitted	Dual-business personalization re-not allowed	K	P	Re-personalization cannot be performed on a dual-business cash register.
HTTP 422 Unprocessable Content	B2025	Owner change to current taxpayer not permitted	Tax number conflict, re-personalization not allowed	K	P	The e-cash register cannot be re-personalized to the current taxpayer. The process must be restarted with a commissioning code belonging to a different taxpayer.
HTTP 422 Unprocessable Content	M0010	DECRYPTION_ERROR	The submitted receipt cannot be decrypted	K	P	The e-cash register must submit the symmetric key used for encryption in the expected format and use the designated encryption algorithm.
HTTP 422 Unprocessable Content	M0011	FILETYPE_ERROR	Decrypted data is not in GZIP format	K	P	The decrypted data must be in GZIP format.
HTTP 422 Unprocessable Content	M0012	DECOMPRESSION_ERROR	Error occurred during GZIP decompression	K	P	GZIP compression must be performed according to its specification, and the entire compressed file must be encrypted.
HTTP 403 Forbidden	B0001	IP address error	IP address issue	Sz	T	The request must be retried at a later time. At least 30 minutes must pass between the original and repeated requests.
HTTP 403 Forbidden	M0004	AP number conflict auth cert	The AP number in the request does not match the one in the authentication certificate	K	P	The request message must contain the same AP number as in the client certificate.
HTTP 403 Forbidden	M0005	Tax number conflict auth cert	The tax number in the request does not match the one in the	K	P	The request message must contain the same tax number as in the client certificate.



HTTP Response	Error Code	Message	Error Reason	Type	Nature	Explanation / Action
			authentication certificate			
HTTP 403 Forbidden	M0007	AP number conflict signing cert	The AP number in the request does not match the one in the signing certificate	K	P	The request message must contain the same AP number as in the signing certificate used for signing the submitted receipt envelope.
HTTP 403 Forbidden	M0008	Tax number conflict signing cert	The tax number in the request does not match the one in the signing certificate	K	P	The request message must contain the same tax number as in the signing certificate used for signing the submitted receipt envelope.
HTTP 422 Unprocessable Content	B0004	Record counter not unique	The recordCounter in the request is not unique	K	P	The recordCounter provided for the given AP number and taxpayer has already been submitted, and no process identifier for repeated submission has been specified. Either a new recordCounter must be provided, or a process identifier must be included.
HTTP 422 Unprocessable Content	B0005	Invalid last record counter	The lastRecordCounter in the request is incorrect	K	P	The lastRecordCounter in the request cannot be smaller than the recordCounter and must not be smaller than the previously submitted lastRecordCounter for the given AP number and taxpayer.
HTTP 422 Unprocessable Content	N/A	End-of-operation not permitted	Termination of operation is not allowed	Sz	T	The given e-cash register does not meet the conditions required for the "End of Operation" service; therefore, termination of operation cannot be executed.
HTTP 422 Unprocessable Content	B2001	Invalid license number	Invalid license number in AP number	K	P	The e-cash register is attempting to register with an AP number containing a non-existent license number. The process can be restarted with a valid AP number containing a correct license number.



11.2.3 Receipt repository response codes

This table contains the response codes that the receipt repository may return to the customer application.

HTTP Response	Error Code	Message	Error Reason	Type	Nature	Explanation / Action
HTTP 403 Forbidden	B5004	Invalid token	The customer application attempted to query using an invalid token.	K	P	The application must perform a new registration.
HTTP 200 OK	B5005	Receipt not found	No receipt found for the given search key.	Sz	T	The request can be retried as described in the "Receipt Query" subsection.
HTTP 403 Forbidden	B5006	INVALID_APP_SOFTW ARE_VERSI ON	The software version provided in the application registration call is incorrect.	K	P	The application must provide a valid, authorized software version.
HTTP 403 Forbidden	B5011	REVOKE APP_SOFT WARE_VE RSION	In the application registration call, the software version belongs to revoked software.	K	P	The application must provide a valid, authorized software version.

11.3 Validation errors and error codes

The ["Receipt Submission"](#) and ["Report reception"](#) interfaces process the BASE64-encoded receipt data and related data in a separate step after accepting the data.

Error Type	Error Code	Error Reason	Action Required
Technical Error	SCHEMA_VIOLATION	Non-schema-valid XML	The submitted XML contains elements that violate XSD constraints (as listed in the response). To correct the error,



			contact the e-cash register distributor.
Technical Error	DECOMPRESSION_ERROR	Error during decompression	The data cannot be decompressed. To correct the error, contact the e-cash register distributor.
Business Error	SUPPLIER_TAX_NUMBER_MISMATCH	The seller's tax number differs from the one in the request	The tax number in the receipt differs from the tax number of the e-cash register operator. To correct the error, contact the e-cash register distributor.

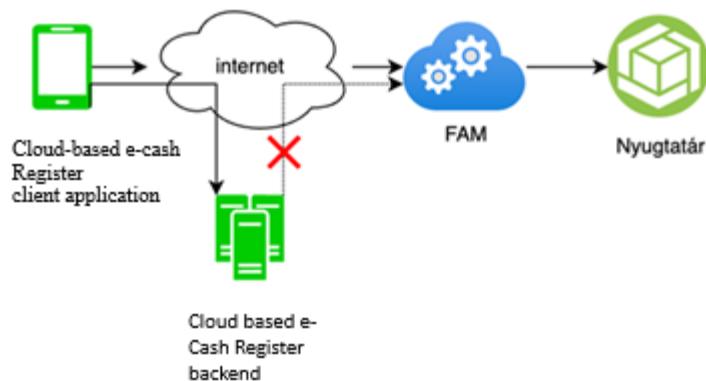
12 Cloud-Based Fiscal Module (FAM)

The Cloud-Based Fiscal Module (FAM) is a service that functions as the cloud-based software version of the Fiscal Unit (AE). Its purpose is to provide an integrated solution for handling fiscal receipts, including arithmetic calculations (e.g., summing line items, generating financial reports) and administrative tasks (e.g., logging, authentication), as well as ensuring data reporting and control for tax authorities.

Key Functions of FAM:

- **Management of virtual fiscal units** – Each FAM instance is uniquely identified by an AP number, storing data and status.
- **Receipt processing** – Handling of tax and non-tax receipts, cash receipts, invoices, simplified invoices, cancellation and modification receipts, cash movement receipts, cash register reports, daily turnover reports, and compilation of individual receipts.
- **Arithmetic and financial calculations** – Summing line items, rounding calculations, tax content calculations, and generating numerical reports.
- **Receipt image data generation** – Preparing data for printed or displayed receipt copies, including headers, line items, barcodes, metadata, and control data, while formatting numbers for minimal client-side processing.
- **Secure data storage** – Logging receipt data and key events in an authentic and verifiable format.
- **Data reporting** – Forwarding receipts to the receipt repository (Nyugtatár).
- **Receiving and executing central commands** – Processing and executing NAV-I system instructions, such as updating taxpayer data, VAT rates, suspending operations, blocking, or unblocking the system.

FAM is a mandatory fiscal backend service for cloud-based e-cash register applications. Applications may connect to their own backend for additional business services, but all fiscal functions must be performed exclusively through FAM, with direct integration. FAM cannot be accessed from an e-cash register's business backend.



Each cloud-based e-cash register has a dedicated virtual fiscal unit, identified by a unique AP number. After certification, each cloud-based e-cash register software is assigned a unique



license number, just like physical cash registers, and each instance is identified by a serial number, e.g., "C12345678"

Every virtual fiscal unit maintains a full state descriptor, and all operational data is stored in the FAM database. It keeps records of:

- Fiscal day status,
- Daily receipt serial numbers,
- Daily and global revenue collectors, etc.

As a result, FAM functions as a virtual cash register, where the e-cash register client application serves as the user interface. Since minimal data storage is required on the client, it is generally sufficient to store an authentication certificate for access.

If a client application needs to be reinstalled or migrated to a new device, the user can continue their operations seamlessly by simply requesting a new access certificate.

FAM operates via microservices architecture, where its API endpoints perform atomic operations on the FAM instance managed by the client application. Each request is considered executed once FAM confirms it in the response message.

12.1 Registration service

The cloud-based e-cash register application can only be used by authorized natural or legal persons; therefore, the cloud-based e-cash register application request must be submitted on the e-cash register portal provided by NAV.

Logging into the registration portal requires KAÜ authentication, and the portal verifies through an external service whether the logged-in user is authorized to represent the specified natural or legal person.

During the request, the user selects which cloud-based e-cash register application they wish to use. After verifying the authorization, the portal initiates the creation of a virtual fiscal unit instance in FAM, which receives an AP number corresponding to the selected software's license number. This process executes a standard cash register registration process towards NAV-I. The registration portal automatically requests the commissioning code.

The portal displays the AP number of the new virtual fiscal unit instance, the technical user credentials, and a one-time-use token in a machine-readable format (QR code). After downloading the selected application (and optionally registering it with the manufacturer), the QR code must be scanned into the application, which generates a client certificate request. This request is sent to the FAM client authentication certificate request endpoint using the authentication data read from the QR code.

After the certificate is created, the technical user must log into FAM, resulting in the application receiving a long-term session token.

After issuing the certificate, its keys and the session token must be stored in the protected storage of the operating system and must be secured with the PIN code required in the application. The PIN code request can optionally be replaced by biometric authentication provided by the device.



After this, setting up a printer for the e-cash register client application is mandatory. This can be done via wired or wireless (e.g., Bluetooth) connection. The printer's functionality must be tested by printing a test page.

With the client authentication certificate and session token, and after setting up the printer, the cloud-based e-cash register client sends a “hello” message, after which the application’s fiscal functions can begin.

Summary of the main registration steps:

- Request on the ePG portal
 - Verification checks
 - Assignment of FAM instance, NAV registration
 - Returning activation data
- Application Setup
 - Scanning activation data (QR code)
 - Requesting client certificate
 - Application-level login to FAM
 - Connecting and testing the printer
 - Reporting completion of setup ("hello")

The certificate's validity must be checked at application startup and at least once daily. If the application does not detect the certificate (e.g., due to accidental data deletion), if the user forgets their PIN code, or if the certificate expires, the user can request a new QR code via the e-cash register registration interface and must request a new client authentication certificate using the procedure described above.

The URLs of the FAM service endpoints in the inspection environment differ from those in the live environment, and the e-cash register client must be prepared to handle both environments. The identification of the environments is based on the AP number contained in the activation QR code, as described in the section [Interpretation of Cloud-based e-Cash Register Activation QR Code](#).

12.2 E-Cash Register REST interface – general information

The e-cash register client communicates with FAM via a standard REST API over an SSL/TLS connection. During connection establishment, the client authentication certificate generated during registration is used.

Fiscal function endpoints in FAM can only be accessed by presenting the certificate. The only exception is the certificate (sign) request.

API Endpoint Categories:

- **Authentication** – User login and logout.
- **Document** – Endpoints related to **receipt creation and printing**.
- **Currencies** – Endpoints managing **currencies handled by FAM**.
- **PaymentMethod** – Default **built-in payment methods** (as required by the Cash Register Regulation) and custom payment methods.
- **File** – Interface for **downloading animated GIF and WAV audio files** provided by NAV, indicating successful submission to NAV-I.



- **ConnectionInitializer** – Endpoint for issuing the **fiscal system client certificate** (part of the FAM backend system).
- **System** – Endpoints for **status queries**.
- **Telemetry** – NAV-I communication functions.

FAM service endpoints use one of the following context roots:

- /fam-ca/v1 – Endpoints related to certificate management for FePG applications.
- /fam/v1 – Business endpoints accessed by FePG applications using a client certificate.

HTTP methods

- GET – Retrieves data using parameters passed in the URL.
- POST – Records or modifies data using parameters in the request body.
- DELETE – Deletes data using parameters passed in the URL.

HTTP headers

The HTTP headers used by the application are divided into two categories:

- Mandatory
- Temporary

Header Key	Header Value/Format	Type
Content-Type	application/json	Mandatory
X-API-Version	X	Mandatory
X-Auth-Token	12345678-90abcdef-0123-4567890abcdef	Mandatory
User-Agent	<app name>/<app version> (<OS>; <OS version>; <device>)<platform>/<platform version>	Mandatory

Content-Type:

The Content-Type header indicates the original media type of the resource.

X-API-Version:

The X-API-Version header ensures that the client and the server-side application are synchronized. The server cannot accept or process requests with an incorrect data format. The X-API-Version header allows the client to inform the server about the API version it supports. If the server-side version does not match the version sent by the client, the server will reject the request. FAM is introduced with version number "1".

X-Auth-Token:

The X-Auth-Token HTTP header is only used when a user has an active session. In this case, the login session token must be included in the HTTP headers. If the header does not contain the logged-in user's token but the server expects it, the server will automatically return an HTTP 401 error response to the client.



User-Agent:

The FePG client sends application- and environment-specific data in the User-Agent HTTP header. The header must be dynamically generated from the following data:

- Application Data:
 - Application name – The name registered in the app store
 - Application version – The version number registered in the app store
 - Environment Data:
 - Operating system – "iOS" or "Android"
 - Operating system version (e.g., Build.VERSION.RELEASE on Android, UIDevice.current.systemVersion on iOS)
 - Device – The model identifier of the device running the application (e.g., Build.MODEL on Android, UIDevice.current.model on iOS)
- Platform Data:
 - Platform – The official name of the application development framework
 - Platform version – The version of the framework in which the given application version was built

The User-Agent HTTP header follows this format:<app name>/<app version> (<os>; <os version>; <device>) <platform>/<platform version>

For example:

- "NAV FePG/1.0 (iOS; 18.1.1; iPhone14,2) ExoPlayerLib/2.11.4"
- "NAV Vevői App/1.0 (Android; 13.0; Pixel 7) ExoPlayerLib/2.9.0"

Serialization:

Incoming and outgoing objects are serialized in JSON format. In FAM responses, fields with null values are optional.

12.2.1 General request validations and response messages

FAM performs validation on all incoming requests and processes only those calls whose data content is complete and complies with the rules of the given endpoint.

The prohibited characters in text fields are „<”, „>”, „{”, „}”, „[”, „]”, „\$” and „\” except for the line break markers „\n”, „\r” and „\t”.

The response messages for endpoint calls always contain the following fixed fields:

- **resultCode** The identifier code of the task result
- **resultDesc** - A short textual description of the task result, including error messages.

For a successful endpoint call, the response will include resultCode = SUCCESS. Other success-related response codes are detailed in the descriptions of the individual endpoints.

For every request, the following validations are checked. If any of these validations fail, FAM returns the corresponding result code in an **HTTP 200 response**. If a particular endpoint call does not require one of these validations, it is explicitly mentioned in the respective endpoint description.



If an error code is returned in the response message, it is sufficient to read and interpret only the resultCode and resultDesc fields. In some exceptional cases, additional details may be provided, which will be highlighted in the relevant sections.

Each **endpoint group** highlights the general validations and error codes applicable to that group. The specific validations, error codes, and actions related to individual endpoints are detailed in their respective descriptions.

Validation/Description	Result Code (resultCode)	J	Action Required
API version check. Error: The X-API-Version HTTP header parameter does not match the expected version.	API_VERSION_MISMATCH	P	Provide the correct X-API-Version in the HTTP header.
Correct completion of mandatory parameters. Error: One or more required parameters are missing from the request or contain prohibited characters. The missing or prohibited parameter is returned in the resultDesc field of the response message. The content of the resultDesc field designates only one mandatory field to be completed even if multiple mandatory fields are missing from the request or if it contains prohibited characters.	MISSING_REQUEST_PARAMETERS	P	Ensure all required fields are filled in.
AP number (systemId) verification. Error: The AP number (systemId) provided in the request does not match the one in the application's client certificate.	INVALID_SYSTEM_ID	P	Provide the correct AP number.
Blocked state check. Error: The cash register (FAM instance) is in a blocked state.	FCU_IS_BLOCKED	T	The e-cash register operator can check the reason for the block on the ePG Portal or inquire with NAV. In a blocked state, only the following FAM interfaces can be accessed: Client authentication certificate endpoints Login Cash register information and event submission FAM instance status query Peripheral settings and query Telemetry (except for the domestic tax number query endpoint)

The **nature of the error** (J column) can be:

- **Permanent (P)** – These errors must be reported to operations/developers. Resolving these errors requires **operational intervention and/or software version modification**.
- **Transient (T)** – These errors occur **only in the current state**. They can be resolved **through normal business intervention or state modification** without requiring a software version update. (*For example, a cash register in a blocked state can be reactivated through a business process.*)



12.2.2 Key Data Fields (and their value sets)

This section describes key data fields that affect the entire system and/or have predefined value sets.

12.2.2.1 systemId – AP umber

The unique identifier of the FAM instance (AP number).

- 9-character text field: 1 letter and 8 numbers
- If this data appears in a request, it must always be provided (*).

12.2.2.2 documentId / docId – FAM document identifier

documentId or docId – The unique identifier of the related document within the FAM system.

This is NOT the receipt identifier, which is stored in the **docNo** field.

12.2.2.3 fiscalDayNo

The sequential number of the fiscal day.

12.2.2.4 Representation of monetary values and quantities

FAM monetary values and quantity fields are stored as strings and their maximum length is 26 numeric characters. For decimal fractions, a decimal point (.) must be used

12.2.2.5 Document type

The document type is used in multiple fields:

- openDocuments/type
- printSpool/type
- documentDescriptor/type
- docDesc/type
- documentType

Possible values:

- **CASH_FLOW_REPORT** - Cash movement document
- **CASH_REGISTER_REPORT** - Cash register report
- **CUSTOM_DOC** – Custom document
- **FISCAL_DAY_OPEN** - Fiscal day opening document
- **FISCAL_DAY_REPORT** - Daily turnover report
- **INVOICE** - Invoice
- **RECEIPT** - Receipt
- **RECEIPT_LIST_REPORT** - Receipt summary report
- **RETURN_RECEIPT** - Corrective document
- **SIMPLE_INVOICE** – Simplified invoice
- **VOID_RECEIPT** - Cancellation document

12.2.2.6 addressType - Address detail level

- **SIMPLE** – Domestic VAT taxpayer
- **DETAILED** – Non-VAT taxpayer, natural person.

The completeness of the related data structure depends on this field's value. More details are available in the address object description [taxpayer objects](#).



12.2.2.7 customerVatStatus - VAT status of the customer.

Possible values:

- **DOMESTIC** – Domestic VAT taxpayer
- **PRIVATE_PERSON** – Non-VAT taxpayer, natural person
- **OTHER** – Other

12.2.2.8 ecrState - Cash register status

Possible values:

- **OK** – The cash register is operating normally
- **BLOCK** – The cash register is in a blocked state
- **ERROR** – A malfunction has occurred in the cash register

12.2.2.9 fcuState - FAM instance status

Possible values:

- **NONE** – Technical status indicator for non-existent instances
- **CREATED** - Initialized FAM instance
- **WAITING_FOR_CERT** - Instance is registered in NAV-I but awaiting certificates
- **PENDING** - Registration complete, waiting for the cash register
- **REGISTERED** - Commissioned, successful connection with the cash register (cash register successfully called the "hello" telemetry call)
- **SUSPENDED** - Operation suspended

12.2.2.10 invoiceType – Invoice type

Possible values:

- **ELECTRONIC** - Electronic
- **PAPER** – Paper based

12.2.2.11 role – user role

The user role registered in the FAM system (cash register).

Possible values:

- **ROLE_ADMIN** – The default (and only) role in FAM

12.2.2.12 (TaxRates)/type / temporalType – data validity type

Possible values:

- **OLD** – Previous/Expired
- **CURRENT** - Actual
- **NEXT** – Next/Upcoming

12.2.2.13 moneyCat / moneySubCat Payment methods and subcategories

Payment Method Types (**moneyCat**):

- **CASH** - Cash
- **CARD** – Bank card
- **SZEP** – SZÉP card



- **AFR** – Instant Payment System
- **OTHER_CHANGE_RETURNABLE** - Other payment method with change given
- **OTHER_CHANGE_NON_RETURNABLE** - Other payment method without change given
- **CHANGE** - Change (Note: calculated by FAM and returned in the HTTP response; should not be included in the request.)
- **ROUND** - Rounding (Note: calculated by FAM and returned in the HTTP response; should not be included in the request.)
- **WIRE_TRANSFER** - Bank transfer (interpreted only in the case of an invoice)

Payment Method Subcategories (**moneySubCat**):

- **AJÁND** – Gift voucher
- **HŰSÉG** - Loyalty card
- **SMART** - Smartcard
- **GÖNGY** - Returnable deposit slip
- **KUPON** – Coupon
- For other cases, the **name of the payment method** should be stored.

12.2.3 Key object descriptions

This section contains key descriptors that appear in multiple locations within the system and messages, maintaining the same field structure.

12.2.3.1 address - object

```
"< block_identifier_name>": {  
    "address": {  
        "addressType": "DETAILED",  
        "countryCode": "HU",  
        "region": null,  
        "postCode": "1500",  
        "city": "Géc",  
        "additionalAddressDetail": null,  
        "streetName": "Lik",  
        "publicPlaceCategory": "köz",  
        "houseNumber": "42",  
        "building": null,  
        "staircase": null,  
        "floor": "3",  
        "door": "5",  
        "lotNumber": null  
    },  
    ...  
},
```

Data structure field explanation:

* Fields marked with * are mandatory

- **(DETAILED) address** – Full address data
 - **addressType*** – Address detail level
 - DETAILED – Detailed address
 - **countryCode*** – ISO 3166 country code
 - **region*** – ISO 3166 region code



- **postCode*** – Postal code
 - **city*** – City
 - **streetName*** - Street name
 - **publicPlaceCategory*** - Type of public place
 - **houseNumber** – House number
 - **building** – Building
 - **staircase** – Staircase
 - **floor** – Floor
 - **door** – Door
 - **lotNumber** - Parcel number
 - **additionalAddressDetail*** – Additional address details
-
- **(SIMPLE) address** – Simplified Address Data
 - **addressType*** – Address detail level
 - SIMPLE – Simplified address
 - **countryCode*** – ISO 3166 country code
 - **region*** – ISO 3166 region code
 - **postCode*** – Postal code
 - **city*** – City
 - **additionalAddressDetail*** – Additional address details

12.2.3.2 documentDescriptor – object

The documentDescriptor object contains the **constant data and structure** of document records. In some cases, additional data may be included (e.g., printSpool → documentData for receipt images).

```
"<block_identifier_name>": [  
  {  
    "type": "SIMPLE_INVOICE",  
    "fiscalDayNo": 2,  
    "docId": 5,  
    "interrupted": false  
  }  
,
```

Data structure field explanation:

* Fields marked with * are mandatory

- **documentDescriptor** - Data structure identifying the document
 - **type*** - Document type
The possible values are listed in the [Document Type](#) section.
 - **docId*** Unique identifier of the related document (within the FAM system).
 - **fiscalDayNo*** - Fiscal day number
 - **interrupted*** - Boolean flag indicating whether the document was interrupted.

12.2.3.3 taxpayer - object

The taxpayer object contains **constant data and structure** for taxpayer information. In some cases, additional data may be included (e.g., **address data** via the address object).



```
"taxpayer": {  
    "address": {  
        "addressType": "DETAILED",  
        "countryCode": "HU",  
        "region": null,  
        "postCode": "1500",  
        "city": "Géc",  
        "additionalAddressDetail": null,  
        "streetName": "Lik",  
        "publicPlaceCategory": "köz",  
        "houseNumber": "42",  
        "building": null,  
        "staircase": null,  
        "floor": "3",  
        "door": "5",  
        "lotNumber": null  
    },  
    "taxpayerName": "Kis Miska",  
    "taxpayerShortName": null,  
    "taxNumber": {  
        "taxpayerId": "20000002",  
        "vatCode": "2",  
        "countyCode": "22"  
    }  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **taxpayer** – taxpayer data
 - **taxpayerName** – taxpayer full name
 - **taxpayerShortName** – taxpayer short name
 - **taxNumber** – tax number

Further related data structure breakdown can be found in the next section under the taxNumber field.

12.2.3.3.1 taxNumber - object

- **taxNumber**
 - **taxPayerId** – taxpayer id
 - **vatCode** – VAT code
 - **countyCode** – County code

12.3 User authentication

FAM identifies the user at both the data connection level and the application level. The first authentication level is the client authentication certificate, which is generated by the application based on the registration data obtained via the QR code on the e-cash register registration interface.

Once the certificate is issued, the user must also log in to the application, using the credentials contained in the registration QR code.



Registration QR Code Contents:

- AP number, e.g., C12345678
- Username, which is identical to the AP number
- Password, a generated character sequence
- Token, a character sequence required for generating the client authentication certificate. This token is **valid for 15 minutes** after QR code generation.

12.3.1 Requesting a client authentication certificate

The user has 15 minutes to request a client authentication certificate after the QR code is generated. The necessary request data is included in the QR code.

API endpoint group details: FAM interface/ConnectionInit

Endpoint Component: ConnectionInitController

HTTP Method: POST

Context Root: /fam-ca/v1

Endpoint URL: /ci/sign

Endpoint Request Objects: SignRequest

Endpoint Response Objects: SignResponse

Request data structure:

```
{  
    "systemId": "C12345678",  
    "username": "C12345678",  
    "password": "656789",  
    "token": "65678953426782789643764783467",  
    "csr": " MIIIFSDCCBDCg ..."}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number).
- **username*** – Identical to the systemId (AP number).
- **password*** – The generated password retrieved from the registration QR code.
- **token*** – A time-based one-time password retrieved from the registration QR code.
- **csr*** – The Certificate Signing Request (CSR) in DER format, encoded in Base64.
- **ecrSoftware*** - The e-cash register software data – See the data structure in the [Telemetry](#) section Submitting cash register information.

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "certId": "Btt6rirtr677vvr7iv8"}
```

Explanation of Unique Fields in the Data Structure:

- **certId**– The certificate issuance process identifier. This ID can be used to query the status of the certificate issuance process.



Before processing the request, the system performs general request validations ([General request validations and response messages](#)), except for the following validation check:

- INVALID_SYSTEM_ID
- FCU_IS_BLOCKED

Then, the system performs the **endpoint-specific** validation checks:

Validation/Description	Result Code (resultCode)	J	Action Required
The certificate request is invalid.	CERTIFICATE_REQUEST_INVALID	T	Submit a correctly formatted and valid certificate request.
The provided token has already been used.	TOKEN_ALREADY_USED	T	Request a new QR code.
The provided username, password, token, or systemId is incorrect.	BAD_CREDENTIALS	T	Provide the correct credentials.
The token has expired.	TOKEN_EXPIRED	T	Request a new QR code.
The software version is not supported	UNSUPPORTED_ECR_SOFTWARE	P	Update eCash register software
Successful but the software version is not the latest.	SUCCESS_APP_VERSION_OLD	P	Update eCash register software

12.3.2 Client authentication certificate renewal

If the client authentication certificate is still valid but has less than 30 days until expiration, the client application must automatically request renewal.

To renew the certificate, the client must:

1. Generate a new certificate signing request (CSR) in DER format.
2. Submit the CSR wrapped in a CMS SignedData format, signed with the existing valid certificate.

If the certificate has already expired, the client must request a new certificate before reconnecting to FAM. The new certificate request must be made via the ePG Portal.

The CMS SignedData format is described in RFC 5652.

API endpoint group details: FAM interface/ConnectionInit

Endpoint Component: ConnectionInitController

HTTP Method: POST

Context Root: /fam-ca/v1

Endpoint URL: /ci/renew

Endpoint Request Objects: RenewRequest

Endpoint Response Objects: RenewResponse

Request data structure

```
{  
    "systemId": "C12345678",  
    "cms": "iVBORw0KGgoAA.../GQWN8AAAAAE1FTkSuQmCC",  
    "ecrSoftware": {  
        ...  
    }  
}
```



Data structure field explanation:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)
- **cms*** – CMS with base64 coding.
- **ecrSoftware*** - The e-cash register software data – See the data structure in the [Telemetry](#) chapter, section Submitting cash register information.

Response Data Structure (Upon Successful Execution):

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "certId": "Btt6rirtr677vvr7iv87o8n7n8o78"  
}
```

Data structure field explanation:

- **certId** – The certificate issuance process identifier, which can be used to query the status of the process

The system first performs general request validations ([General request validations and response messages](#)), except for the following validation checks:

- INVALID_SYSTEM_ID
- FCU_IS_BLOCKED

Then, it performs the **endpoint-specific** validation checks:

Validation/Description	Result Code (resultCode)	J	Action Required
The certificate request is invalid.	CERTIFICATE_REQUEST_INVALID	T	Submit a correctly formatted and valid certificate request.
The software version is not supported	UNSUPPORTED ECR SOFTWARE	P	Update eCash register software
Successful but the software version is not the latest.	SUCCESS_APP_VERSION_OLD	P	Update eCash register software

12.3.3 Client authentication certificate download

The certificate issuance process starts with a certificate signing or renewal request. The process status must be polled repeatedly.

- As long as the **resultCode** in the response is **IN_PROGRESS**, the endpoint must be called again every 5 seconds.
- If the resultCode changes to **SUCCESS**, the certificate has been generated, and the certificate field will be populated.
- If after 2 minutes of polling the response does not change to **SUCCESS**, the process must be considered failed.

API endpoint group details: FAM interface/ConnectionInit

Endpoint Component: ConnectionInitController

HTTP Method: GET

Context Root: /fam-ca/v1

Endpoint URL: /ci/query-cert/download?certId={certId}

Endpoint Response Objects: QueryCertResponse



Request data structure

For the **GET** method, no request body is required.

The URL parameter from the request:

* Fields marked with * are mandatory

- **certId*** - The certificate issuance process identifier.

Data structure of the response

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "certificate": " MIIFSDCCBDCg..."  
}
```

Data structure field explanation:

- **resultCode** - The identifier code for the task result:
 - **SUCCESS** - The certificate has been issued
 - **IN_PROGRESS** The certificate issuance is in progress
 - **ERROR** - Unknown certId
- **certificate** – The issued client authentication certificate in DER format, encoded in Base64.

The system first performs general request validations, then executes endpoint-specific checks ([General request validations and response messages](#)):

Validation/Description	Result (resultCode)	Code	J	Action Required
Invalid certificate identifier.	ERROR	T		Provide a valid certId in the request URL.

12.3.4 Login

To use the FAM's fiscal functions, login must be performed using the technical administrator user of the FAM instance.

- The technical username for login is the AP number.
- The password is found in the QR code displayed on the registration interface (inside the password field).

Upon successful login, the returned session token must be included in the X-Auth-Token field of the HTTP header for all REST calls that require authentication.

- The session token has multi-year expiration, so repeated logins are not necessary.
- The authentication client certificate, technical username and password, and the session token must be stored securely in the protected storage of the operating system.

API endpoint group details: FAM interface/Authentication

Endpoint Component: Authentication Controller

HTTP Method: POST

Context Root: /fam/v1



Endpoint URL: /auth/login

Endpoint Request Objects: LoginRequest

Endpoint Response Objects: LoginResponse

Request data structure:

```
{  
    "systemId": "C12345678",  
    "username": "C12345678",  
    "password": "P455W0rD"  
}
```

Data structure field explanation:

- **systemId*** – The unique identifier of the FAM instance (AP number)
- **username*** The technical username, which is the AP number of the FAM instance
- **password*** The technical user password, retrieved from the registration QR code

Response Data Structure (Upon Successful Execution):

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "realUserName": "",  
    "role": "ROLE_ADMIN"  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **realUserName** The full name of the user, which is an empty string ("") in FAM
- **role** – The technical user role, which is always "ROLE_ADMIN"

The **session token** is returned in the **X-Auth-Token** field of the response message.

The system first performs general request validations ([General request validations and response messages](#)) except for the following validation check:

- INVALID_SYSTEM_ID

Then, it performs **endpoint-specific** validation checks

Validation/Description	Result Code (resultCode)	J	Action Required
Invalid login credentials.	BAD_CREDENTIALS	T	Provide correct credentials in the request data structure.
The user has exceeded the maximum number of failed login attempts.	TOO_MANY_FAILED_LOGIN_ATTEMPTS	T	Wait for the expiration period, then try again.
The systemId in the certificate does not match the one in the login request.	CERTIFICATE_MISMATCH	P	Ensure that the correct systemId and password are provided in the request.



12.3.5 Logout

During normal operation it is not necessary, but in certain situations it may be required to log the client out of the FAM.

After the endpoint call, the FAM deletes the session, the session token stored in the client application becomes invalid, and from then on only the FAM login endpoint can be called by submitting the stored username (AP number) and login password in the application.

API endpoint group details: FAM interface/Authentication

Endpoint Component: Authentication Controller

Endpoint HTTP method: GET

Context Root: /fam/v1

Endpoint URL: /auth/logout

Endpoint response objects: LogoutResponse

Response Data Structure (Upon Successful Execution):

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": ""  
}
```

12.4 State management

The operation of a FAM instance is defined by the following attribute sets:

- Instance State – Represents the status of the instance within the system.
- FAM Instance Operational Parameters – Define the instance's configuration.
 - Usage Parameters – Attributes that influence the operation of the FAM, such as whether the fiscal day is open or the current receipt counter value.
 - Controlled Parameters – These parameters are assigned to the FAM instance during status changes (e.g., system initialization), through tax authority control or inherited from central business parameters.
 - User Parameters – These include settings that can be configured from the cash register.

Changes in controlled parameters (e.g., taxpayer data, VAT rates, etc.) must be retrieved by the cash register at defined intervals or when specific events occur (e.g., closing of the fiscal day). The cash register must also apply these changes within its own operational logic.

It is critical that the cash register accurately tracks the status of the FAM instance. To support this:

- FAM provides dedicated endpoints for querying state descriptor parameters.
- FAM validates each process initiated by the cash register to determine if the requested operation is allowed in the current status.
- FAM sends feedback in the response message, indicating whether the requested operation can be performed.

Based on Its System Status, a FAM Instance Can Have One of the Following States (as Defined in the fcuState Field):

- NONE – Technical status indicator for non-existing instances.
- CREATED – Status indicator for an initialized FAM instance.



- WAITING_FOR_CERT – The instance is registered in NAV-I but is still waiting for certificates.
- PENDING – Registration is complete, waiting for the cash register.
- REGISTERED – The instance is deployed, and a successful connection has been established with the cash register (the cash register has successfully called the "hello" telemetry request).
- SUSPENDED – The instance operation is suspended.

This sequence represents the chronological order of states, with the only allowed transition back being from SUSPENDED to REGISTERED.

The PENDING state is reached at the end of the application request process initiated on the e-cash register registration portal, before the QR code with login credentials is displayed.

A FAM instance is only capable of executing fiscal operations in the REGISTERED state.

12.4.1 Querying the FAM instance status

The FAM interface provides dedicated endpoints for querying the FAM instance status. The purpose of this query is to synchronize the client and server states.

- The entire internal state of the FAM (relevant for the cash register) can be retrieved in a single request.
- However, since this returns a large amount of data, FAM also provides a status-checking endpoint that only indicates if the FAM status has changed since the last query.
- This reduces network load and allows more frequent state checks.

The e-cash register client must perform the status query:

- At every startup
- Whenever the application is brought to the foreground
- Periodically

System-level and document-related status queries can also be performed using the /system/state and /doc/state endpoints (explained later in the document). If these endpoints are used, the /system/status endpoint query can be skipped, as the two other endpoints provide complete functionality.

The Status Descriptor Returns the Following Key Data (The full data structure is detailed in the endpoint descriptions):

- FAM Identifiers (e.g., "NAV", software version)
- FAM Instance Status (one of "NONE", "CREATED", "WAITING_FOR_CERT", "PENDING", "REGISTERED", or "SUSPENDED")
- FAM Execution Mode, which is always "CLOUD"
- Operational Parameters:
 - Taxpayer Data and changes
 - VAT Rates and changes
 - Blocked Status



- Print Queue (all mandatory documents for printing)
- Fiscal Day Data (fiscal day number, open receipts)
- Logged-in User

12.4.1.1 Status query endpoint

The status query endpoint can be called in any state of the FAM instance that corresponds to a valid system status. The returned data corresponds to the current status of the instance.

API endpoint group details: FAM Interface / Telemetry - System Functions

Endpoint Component: SystemController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /system/status/{systemId}

Endpoint Response Objects: FcuStatus

Request data structure:

For the GET method, no request body is required

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId** * – The unique identifier of the FAM instance (AP number)

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "now": 1720692472798,  
    "zoneId": "Europe/Budapest",  
    "fcuState": "REGISTERED",  
    "currentOperatorSite": {  
        "effectiveDate": 1720519280905,  
        "temporalType": "CURRENT",  
        "shop": {  
            "address": {  
                "addressType": "DETAILED",  
                "countryCode": "HU",  
                "region": null,  
                "postCode": "1500",  
                "city": "Géc",  
                "additionalAddressDetail": null,  
                "streetName": "Lik",  
                "publicPlaceCategory": "köz",  
                "houseNumber": "42",  
                "building": null,  
                "staircase": null,  
                "floor": "3",  
                "door": "5",  
                "lotNumber": null  
            },  
            "shopName": "Talicska bolt Kft",  
            "shopShortName": "Talicska bolt Kft"  
        },  
        "taxpayer": {  
    }}
```



```
"address": {
    "addressType": "DETAILED",
    "countryCode": "HU",
    "region": null,
    "postCode": "1500",
    "city": "Géc",
    "additionalAddressDetail": null,
    "streetName": "Lik",
    "publicPlaceCategory": "köz",
    "houseNumber": "42",
    "building": null,
    "staircase": null,
    "floor": "3",
    "door": "5",
    "lotNumber": null
},
"taxpayerName": "Kis Miska",
"taxpayerShortName": null,
"taxNumber": {
    "taxpayerId": "20000002",
    "vatCode": "2",
    "countyCode": "22"
}
},
"nextOperatorSite": null,
"currentTaxRates": {
    "type": "CURRENT",
    "effectiveDate": 1720519280905,
    "taxDepartment": [
        {
            "depCode": "A",
            "taxRate": "5",
            "taxPercentage": "4.76",
            "depLabel": "04,76%"
        },
        {
            "depCode": "B",
            "taxRate": "18",
            "taxPercentage": "15.25",
            "depLabel": "15,25%"
        },
        {
            "depCode": "C",
            "taxRate": "27",
            "taxPercentage": "21.26",
            "depLabel": "21,26%"
        },
        {
            "depCode": "D",
            "taxRate": "0",
            "taxPercentage": "0",
            "depLabel": "AJT"
        },
        {
            "depCode": "E",
            "taxRate": "0",
            "taxPercentage": "0",
            "depLabel": ""
        }
    ]
}
```



```
        "depLabel": "AM"
    }
],
},
"nextTaxRates": null,
"locale": {
    "countryCode": "HU",
    "currencies": [
        {
            "currencyCode": "HUF",
            "conversionValue": "1",
            "displayPrecision": 0,
            "print": true,
            "native": true
        }
    ],
    "amountPrecision": 4
},
"openedFiscalDayNo": 1,
"openDocuments": [],
"printSpool": [],
"logo": null,
"mediaPackageId": null,
"username": "admin",
"userRole": "ROLE_ADMIN",
"fcuManufacturer": "NAV",
"softwareVersion": "0.4.0",
"fcuType": "CLOUD",
"lastReceiptNo": 1,
"printMessage": null,
"realUserName": "",
"fiscalDayOpen": true,
"online": true,
"blocked": true,
"blockReasons": [
    "NTCA"
]
}
```

Data structure field explanation:

- **fcuManufacturer** – constant value: „NAV”
- **fcuState** – The status of the FAM instance. (The list of possible values can be found in the fcuState section) [fcuState](#).
- **fcuType** – Always "CLOUD" indicating cloud operation mode
- **now** - The current UTC system time of the FAM at the moment of response, in Unix time format with millisecond precision
- **zoneId** - The server's time zone, represented as a textual name from the "tz database", e.g., "Europe/Budapest"
- **softwareVersion** – The version of the FAM software.
- **blocked** - Indicates if the FAM instance is blocked
 - true – The instance is blocked.
 - false – The instance is not blocked.



- **blockReasons** – If blocked, this field contains the reason(s) for blocking, otherwise null. Possible values:
 - NTCA – The NAV has blocked the e-cash register
 - ECR – The cash register requested the block (e.g., due to technical failure). Not relevant for FAM.
 - OFFLINE – There has been no connection to the NAV system for 72 hours.
 - EXPIRED_AUTH_CERT – The client authentication certificate has expired. Not relevant for FAM.
 - EXPIRED_SIGN_CERT – The FAM instance's signing certificate has expired.
- **currentTaxRates** - A TaxRates object containing the currently active VAT rates.
 - **type** (= "CURRENT") - (Defines the data's validity type.)
Possible values described in section [\(TaxRates\)/type / temporalType](#).
 - **effectiveDate** - The effective date of the VAT rates
 - **taxDepartment** - Details of the tax departments
 - **depCode** - Tax department identifier
 - **taxRate** - Tax rate label
 - **taxPercentage** - VAT rate in percentage
 - **depLabel** - VAT content percentage
- **nextTaxRates** - A TaxRates object containing future VAT rates that will come into effect on a specific date. (Type = "NEXT".)
- **currentOperatorSite** - An OperatorSite object containing current business and owner details:
 - **temporalType** (= "CURRENT") – Defines the data's validity type, possible values are defined in section [\(TaxRates\)/type / temporalType](#).
 - **effectiveDate** - The effective date of the taxpayer data
 - **shop** – Business information
 - **shopName** - Full name of the shop
 - **shopShortName** – Short name of the shop
 - **address** – Operating location address details. (Possible data structure details are described in the [address section](#).)
 - **taxpayer** – Taxpayer details. (Defined in the [taxPayer](#) object section.) Extended with the following details:
 - **address** – Taxpayer address details (Possible data structure details are described in the [address section](#).)
- **nextOperatorSite** - An OperatorSite object containing the future business and owner details that will be valid from a specified future date. (Type = "NEXT".)
- **logo** - The identifier of the logo graphic to be printed in the receipt header.
- **printSpool** - A list of pending receipts awaiting printing (currently limited to one item). (Defined in the [documentDescriptor](#) section.)
- **fiscalDayOpen** - Indicates whether a fiscal day is open in FAM (Boolean)
 - true – A fiscal day is currently open.
 - false – No fiscal day is open.
- **locale** - A Locale object containing regional settings:
 - **amountPrecision** - The precision for decimal calculations



- **countryCode** - Two-letter country code
- **currencies** - An array of used currencies, including the home currency (exchange rate, three-letter currency code., display precision, Whether the final amount must be printed in this currency, indicator of the home currency) (Defined in the “[Currency management](#)” response structure section under the response data structure’s currency point.)
- **online** - (Boolean) Indicates whether the FAM instance is connected to NAV
 - true – FAM is online and connected to NAV.
 - false – No connection to NAV (Nyugtatár).
- **openDocuments** - A list of currently open receipts in FAM. (Defined in the documentDescriptor [documentDescriptor](#) section.)
- **openedFiscalDayNo** - The sequence number of the currently open fiscal day
- **username** - The logged-in user’s name, always identical to the AP number.
- **realUserName** – Always an empty string ("")
- **userRole** - The user role, always "ROLE_ADMIN" (Administrator).
- **mediaPackageId** - The ID of the current audiovisual file, used in the File API
- **printMessage** - A message from NAV to be printed on the receipt
- **lastReceiptNo** - The receipt number of the last issued receipt within the fiscal day.

12.4.1.2 Querying Status Changes

To avoid querying the entire state space every time, the FAM provides a simple status-checking endpoint that indicates whether any attribute or object has changed since the last status query.

The change detection covers all fields of status, except for the current system time ("now").

The cash register only needs to call this endpoint regularly. A full status query is only necessary if a change has been detected.

API endpoint group details: FAM interface/ Telemetry - System Functions

Endpoint Component: SystemController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /system/status-check/{systemId}/{timestamp}

Endpoint Response Objects: FcuStatusCheck

Request data structure

For the **GET** method, no request body is required.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number).
- **timestamp*** - The "now" value received in the most recent FAM status query by the cash register. The FAM uses this to determine whether any status descriptor parameters have been updated since then.



Response data structure in case of unsuccessful execution

The FAM always returns "SUCCESS" and **changed = true** even if the timestamp is not recognized by the FAM as a valid status change time. This simplifies cash register logic, as the cash register can automatically request a full status update if changed = true.

```
{  
    "changed": true,  
    "now": 1621417133141,  
    "resultCode": "SUCCESS",  
    "resultDesc": ""  
}
```

Data structure field explanation:

- **changed** - Indicates whether the status has changed: **true**
- **now** - The FAM system time in Linux timestamp format with millisecond precision.

12.4.2 Querying the FAM system status

This endpoint allows you to query the system-related statuses from the FAM.

The endpoint call provides a summarized version of the FAM status query, excluding document-related data. This allows the e-cash register client to check for system-level tasks, such as new taxpayer data, blocking status, etc.

API endpoint group details: FAM interface/Telemetry – System functions

Endpoint Component: SystemController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /system/state/{systemId}

Endpoint Response Objects: FcuSystemState

Request data structure

For the **GET** method, no request body is required.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)

Response Data Structure (Upon Successful Execution):

```
{  
    "resultCode": "SUCCESS",  
    "now": 1720694864284,  
    "zoneId": "Europe/Budapest",  
    "fcuState": "REGISTERED",  
    "fcuManufacturer": "NAV",  
    "softwareVersion": "0.4.0",  
    "fcuType": "CLOUD",  
    "currentOperatorSite": {  
        "effectiveDate": 1720519280905,  
        "temporalType": "CURRENT",  
        "shop": {  
            "address": {  
                "street": "1000 Budapest, Kossuth Lajos utca 10",  
                "number": "1000",  
                "zip": "H-1000",  
                "city": "Budapest",  
                "country": "Hungary"  
            }  
        }  
    }  
}
```



```
"addressType": "DETAILED",
"countryCode": "HU",
"region": null,
"postCode": "1500",
"city": "Géc",
"additionalAddressDetail": null,
"streetName": "Lik",
"publicPlaceCategory": "köz",
"houseNumber": "42",
"building": null,
"staircase": null,
"floor": "3",
"door": "5",
"lotNumber": null
},
"shopName": "Talicska bolt Kft",
"shopShortName": "Talicska bolt Kft"
},
"taxpayer": {
    "address": {
        "addressType": "DETAILED",
        "countryCode": "HU",
        "region": null,
        "postCode": "1500",
        "city": "Géc",
        "additionalAddressDetail": null,
        "streetName": "Lik",
        "publicPlaceCategory": "köz",
        "houseNumber": "42",
        "building": null,
        "staircase": null,
        "floor": "3",
        "door": "5",
        "lotNumber": null
    },
    "taxpayerName": "Kis Miska",
    "taxpayerShortName": null,
    "taxNumber": {
        "taxpayerId": "20000002",
        "vatCode": "2",
        "countyCode": "22"
    }
}
},
"nextOperatorSite": null,
"currentTaxRates": {
    "type": "CURRENT",
    "effectiveDate": 1720519280905,
    "taxDepartment": [
        {
            "depCode": "A",
            "taxRate": "5",
            "taxPercentage": "4.76",
            "depLabel": "04,76%"
        },
        {
            "depCode": "B",
            "taxRate": "18",
            "taxPercentage": "17.76",
            "depLabel": "017,76%"
        }
    ]
}
```



```
        "taxPercentage": "15.25",
        "depLabel": "15,25%"
    },
    {
        "depCode": "C",
        "taxRate": "27",
        "taxPercentage": "21.26",
        "depLabel": "21,26%"
    },
    {
        "depCode": "D",
        "taxRate": "0",
        "taxPercentage": "0",
        "depLabel": "AJT"
    },
    {
        "depCode": "E",
        "taxRate": "0",
        "taxPercentage": "0",
        "depLabel": "AM"
    }
],
},
"nextTaxRates": null,
"locale": {
    "countryCode": "HU",
    "currencies": [
        {
            "currencyCode": "HUF",
            "conversionValue": "1",
            "displayPrecision": 0,
            "print": true,
            "native": true
        }
    ],
    "amountPrecision": 4
},
"username": "admin",
"realUserName": "admin",
"userRole": "ROLE_ADMIN",
"online": true,
"blocked": true,
"blockReasons": [
    "NTCA"
],
}
}
```

Data structure field explanation:

- The fields are identical to those described in the chapter on “[Querying the FAM instance status](#)”.



12.4.2.1 Querying the status change of the FAM system

To avoid querying the system state space every time, the FAM indicates at a simple status-checking endpoint whether any attribute or object of the state has changed since the last status query. The change check covers all fields of the state except for the current system time ("now").

It is sufficient for the cash register to call this endpoint regularly, and the system status only needs to be queried if it has changed.

API endpoint group details: FAM interface/Telemetry – System functions

Endpoint Component: SystemController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /system/state/{systemId}

Endpoint Response Objects: FcuSystemState

Request data structure

For the GET method, no request body is required.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)
- **timestamp*** - the "now" value received in the last FAM system status queried by the cash register, based on which the FAM can determine whether any status descriptor parameter has been updated since then.

Response Data Structure (Upon Unsuccessful Execution)

The FAM gives a "SUCCESS" and changed = true response to the status change query even if the FAM does not recognize the timestamp as a status change time. This also simplifies the cash register side logic, since if changed = true, the cash register automatically requests the system status.

```
{  
  "changed": true,  
  "now": 1621417133141,  
  "resultCode": "SUCCESS",  
  "resultDesc": ""  
}
```

Data structure field explanation:

- **changed** - Indication of status change: **true**
- **now** - The FAM system time in Linux timestamp format with millisecond resolution.



12.4.3 Document management status

This endpoint allows querying the FAM for statuses related to document management (open document descriptor, print queue) and closely related states (open fiscal day status, sequence number, etc.).

By calling this endpoint, the e-cash register client can check whether there are pending tasks related to previously recorded or unfinished documents. It is recommended to call this endpoint when the client application is brought to the foreground. This helps prevent issues where the client, due to a differing state, might attempt to create a new document before confirming the printing of a previously prepared paper-based simplified invoice, which could otherwise result in an error.

API endpoint group details: FAM interface/Document – Document management

Endpoint Component: DocumentController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /doc/state/{systemId}

Endpoint Response Objects: FcuDocumentState

Request data structure

For the GET method, no request body is required.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "openedFiscalDayNo": 2,  
    "openDocuments": [  
        {  
            "type": "SIMPLE_INVOICE",  
            "fiscalDayNo": 2,  
            "docId": 5,  
            "interrupted": false  
        }  
    ],  
    "printSpool": [  
        {  
            "type": "SIMPLE_INVOICE",  
            "fiscalDayNo": 2,  
            "docId": 5,  
            "interrupted": false,  
            "documentData": [...]  
        }  
    ],  
    "logo": "1",  
    "lastReceiptNo": 4,  
    "fiscalDayOpen": true  
}
```

Data structure field explanation:

- **openedFiscalDayNo** – The sequential number of the opened fiscal day. If the day is closed (=false), the value returned is -1.
- **openDocuments** – Descriptor of the open document. The related data structure is described under the document [documentDescriptor](#) section.
- **printSpool** - List of documents waiting for printing (currently, it can contain only one element). The related data structure is described under the [documentDescriptor](#) section.
- **documentData** – Data required for rendering the document image. The structure of the document image is described under the “[DocumentData - Receipt Image Data](#)”
- **logo** – The identifier of the logo graphic to be printed in the document header.
- **lastReceiptNo** – The sequential number of the last issued receipt within the fiscal day.
- **fiscalDayOpen** – Boolean value, true if a fiscal day is open, otherwise false.

12.5 Telemetry

Telemetry calls facilitate processes between FAM and NAV-I. The available calls include:

- Sending the **hello** message to finalize initialization,
- Querying domestic tax numbers,
- Sending cash register information and events.

The telemetry interface performs general validation as well as endpoint-specific checks. The specific validation response messages related to the interface are summarized in the following table:

12.5.1 Responses, error codes

Validation/Description	Result code	J	Action required
The FAM is running another process in the background, preventing it from serving the request.	FCU_IS_BUSY	T	The request can be resubmitted at a later time, after waiting at least one minute
Unknown server error	UNKNOWN_ERROR	T	The request can be resubmitted at a later time, after waiting at least one minute. If the issue persists, notify the NAV customer service
The NAV-I central system sent an erroneous response	SERVER_ERROR	T	The request can be resubmitted at a later time, after waiting at least thirty (30) minutes. If the issue persists, notify the NAV customer service.
The FAM cannot reach the NAV-I central system.	CONNECTION_ERROR	T	The request can be resubmitted at a later time, after waiting at least one minute. If the issue persists, notify the NAV customer service.
The registration process has not been completed on the FAM instance	FCU_IS_NOT_REGISTERED	T	Before performing any other operation, the Hello endpoint must be called

12.5.2 Hello

API endpoint group details: FAM interface/Telemetry - Telemetry



Endpoint Component: TelemetryController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /telemetry/hello

Endpoint Request Objects: HelloRequest

Endpoint Response Objects: HelloResponse

Request data structure

```
{  
    "systemId": "{{systemId}}"  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null  
}
```

Specific Result Codes for the Endpoint:

Validation/Description	Result code	J	Action required
The FAM instance is not in a PENDING state; the certificates have not yet arrived,	FCU_IS_NOT_PENDING	T	The request can be resubmitted later, after the certificates have been downloaded.
The FAM instance is already in the REGISTERED state, and the Hello endpoint has already been called.	FCU_IS_REGISTERED_ALREADY	P	Repeated calls to the Hello endpoint are unnecessary and should be avoided.

12.5.3 Domestic Tax Number query

API endpoint group details: FAM interface/Telemetry - Telemetry

Endpoint Component: TelemetryController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /telemetry/query-taxpayer

Endpoint Request Objects: QueryTaxpayerRequest

Endpoint Response Objects: QueryTaxpayerResponse

Request data structure

```
{  
    "systemId": "{{systemId}}"  
    "taxpayerId": "20000002"  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)
- **taxpayerId*** - The VAT core number (8 numeric characters)



Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "infoDate": 1717579454000,  
    "taxpayerValidity": true,  
    "taxpayerData": {  
        "taxpayerName": "Minta Mihály",  
        "taxpayerShortName": Misi,  
        "taxNumber": {  
            "taxpayerId": "20000002",  
            "vatCode": "2",  
            "countyCode": "22"  
        },  
        "incorporationType": "SELF_EMPLOYED",  
        "vatGroupMembership": null,  
        "taxpayerAddressList": [  
            {  
                "taxpayerAddressClass": "HQ",  
                "taxpayerAddress": {  
                    "addressType": "DETAILED",  
                    "countryCode": "HU",  
                    "region": null,  
                    "postalCode": "1111",  
                    "city": "Budapest",  
                    "streetName": "Nincs",  
                    "streetType": "Út",  
                    "number": "1",  
                    "building": null,  
                    "staircase": null,  
                    "floor": "4",  
                    "door": "14",  
                    "lotNumber": null  
                }  
            }  
        ]  
    }  
}
```

Data structure field explanation:

- **infoDate** – The date of the last change of the data
- **taxpayerValidity** – Whether the taxpayer exists and is valid
- **taxpayerData** – Taxpayer data

The description of the basic data structure related to this can be found under the [taxpayer](#) section.

In this response, the taxpayerData is supplemented with additional data compared to what is described under taxpayer:

- **incorporation** – Economic type
- **vatGroupMembership** – The taxpayer's VAT group membership
- **taxpayerAddressList** – List of taxpayer addresses
 - **taxpayerAddressClass** – Type of taxpayer address
 - **taxpayerAddress** – Taxpayer address details

The related data structure description can be found under the [address section](#).



12.5.4 Sending cash register state information

API endpoint group details: FAM interface/Telemetry - Telemetry

Endpoint Component: TelemetryController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /telemetry/ecr-info

Endpoint Request Objects: EcrInfoRequest

Endpoint Response Objects: EcrInfoResponse

Request data structure:

```
{  
    "systemId": "{{systemId}}",  
    "ecrState": "OK",  
    "errors": "...",  
    "ecrSoftware": {  
        "softwareId": "ABCDEFGHIJKLMNPQR",  
        "softwareName": "e-pénztárgép",  
        "softwareMainVersion": "2.1",  
        "softwareOperation": "ONLINE_ANDROID",  
        "softwareHash": "",  
        "softwareLastUpdate": 1717579454000,  
        "softwareDevName": "Fejlesztő Péter",  
        "softwareDevContanct": "peter.fejeszto@ceg.hu"  
    },  
    "ecrPosition": {  
        "latitude": "50,53436",  
        "longitude": "17,464",  
        "altitude": 235  
    },  
    "ecrTimeUpdate": {  
        "oldTime": 1717579454000,  
        "newTime": 1717579457000,  
        "changeDuration": 5,  
        "changeMode": "4"  
    }  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)
- **ecrState*** - The state of the cash register
The possible values can be found in the [ecrState- Cash register status](#) section.
- **errors** - List of error codes generated since the last reported errors
- **ecrSoftware*** - Information about the software running on the cash register
 - **softwareId*** - The program identifier
 - **softwareName*** - The program name
 - **softwareMainVersion*** - The main version of the program
 - **softwareOperation*** - The type of cash register operating system.
 - ONLINE_ANDROID
 - ONLINE_IOS
 - ONLINE_HUAWEI
 - ONLINE_SERVICE
 - **softwareHash*** – The SHA256 hash of the software



- **softwareLastUpdateTime*** – The last update timestamp of the software
- **softwareDevName*** – The name of the software developer
- **softwareDevContact*** – The electronic contact information of the software developer
- **ecrPosition** - The geographical location of the cash register, applicable only for mobile stores.
 - **latitude*** – Geographic latitude in WGS84 standard
 - **longitude*** – Geographic longitude in WGS84 standard
 - **altitude*** – Altitude above sea level in meters
- **ecrTimeUpdate** – This field must be completed if the PTG clock has been adjusted
 - **oldTime*** – Previous time (Unix timestamp in milliseconds)
 - **newTime*** – New time (Unix timestamp in milliseconds)
 - **changeDuration** – If the system did not switch time discretely, the transition duration in seconds
 - **changeMode*** – The method of time adjustment:
 - 1 – GSM synchronization
 - 2 – Service intervention
 - 3 – Modified by the operator
 - 4 – Adjusted to the computer clock for PC-based PTG
 - 5 – Other automatic synchronization

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null  
}
```

12.5.5 Sending cash register information

API endpoint group details: FAM interface/Telemetry - Telemetry

Endpoint Component: TelemetryController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /telemetry/ecr-event

Endpoint Request Objects: EcrEventRequest

Endpoint Response Objects: EcrEventResponse

Request data structure

```
{  
    "systemId": "{{systemId}}",  
    "ecrEventType": "POWER_ON",  
    "ecrEventValue": null  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)
- **ecrEventValue** – The event note
- **ecrEventType*** - PTG event type
 - BLOCK - blocking
 - UNBLOCK - unblocking



- MESSAGE_ACK – The feedback on processing the printable message received from NAV in the status query. If the printMessage field in the status query result contained content to be displayed, the client application must confirm to the FAM that the message has been displayed. This event type serves this purpose. Until the client application provides feedback, the printMessage field will remain populated in every query.
- OTHER_EVENT - reporting of other important events. In this case, the ecrEventValue must be filled in.

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null  
}
```

12.6 Currency management

The FAM instance stores the currencies used by the user for arithmetic operations performed in currencies other than HUF. By default, only the HUF currency is assigned to the instance in the system. Through the interfaces described in this subsection, the list of currencies used by the e-cash register can be edited.

The FAM handles HUF and foreign currencies using the same data structure, with the difference that the native currency cannot be deleted.

12.6.1 Key object descriptions and data fields

12.6.1.1 currency

```
{  
    "currencyCode": "HUF",  
    "conversionValue": "1",  
    "displayPrecision": 0,  
    "native": true,  
    "symbol": "Ft"  
}
```

Data structure field explanation:

- **currencyCode** – The 3-character name of the currency
- **conversionValue** – The exchange rate of the currency to the native currency (monetary amount type)
- **displayPrecision** – The number of decimal places when displayed
- **native** – Indicates the native currency, which is only true for the Hungarian forint (HUF).

12.6.2 Querying a specific currency type

API endpoint group details: FAM interface/Currency – Currency management

Endpoint Component: CurrencyController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /currency/{systemId}/{currencyCode}



Endpoint Response Objects: GetCurrencyResponse

Request data structure

For the GET method, no request body is required.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)
- **currencyCode*** - The identifier of the currency to be queried, its 3-character name.

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "currency": {  
        "currencyCode": "HUF",  
        "conversionValue": "1",  
        "displayPrecision": 0,  
        "native": true,  
        "symbol": "Ft"  
    }  
}
```

Data structure field explanation:

- **currency** – The data structure of the currency object is described in the ["Key Object Descriptors and Data Fields"](#) section.

The system first performs the general request validations ([General request validations and response messages](#)), then carries out the specific checks related to the endpoint:

Validation/Description	Result Code	J	Action Required
Unknown currency	CANNOT_GET_CURRENCY	T	Provide the correct currency identifier.

12.6.3 Querying all currencies

API endpoint group details: FAM interface/Currency - Currency Management

Endpoint Component: CurrencyController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /currency/{systemId}

Endpoint Response Objects: GetAllCurrencyResponse

Request data structure

For the GET method, no request body is required.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** - The unique identifier of the FAM instance (AP number)

Response Data Structure (Upon Successful Execution)



```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "currencies": [  
        {  
            "currencyCode": "HUF",  
            "conversionValue": "1",  
            "displayPrecision": 0,  
            "native": true,  
            "symbol": "Ft"  
        }  
    ]  
}
```

Data structure field explanation:

- **fiscalDayNo** - Serial number of the currently open fiscal day
- **currencies** – List of currencies

The description of the data structure related to each currency matches the currency data structure described in [currency](#) section.

12.6.4 Deleting a currency

API endpoint group details: FCU interface/Currency – Handling currencies

Endpoint Component: CurrencyController

HTTP Method: DELETE

Context Root: /fam/v1

Endpoint URL: /currency/{systemId}/{currencyCode}

Endpoint Response Objects: FcuCurrencyResult

Request data structure

Not applicable in the case of the DELETE method.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)
- **currencyCode*** - The identifier of the currency to be queried, a 3-character name

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "success"  
}
```

The native currency (HUF) cannot be deleted.

The system first performs general request validations, then carries out endpoint-specific checks ([General request validations and response messages](#)):

Validation/Description	Result Code	J	Action Required
------------------------	-------------	---	-----------------



The currency cannot be deleted	CANNOT_DELETE_CURRENCY	T	Provide the correct currency identifier
--------------------------------	------------------------	---	---

12.6.5 Adding or modifying a currency type

API endpoint group details: FAM interface/Currency - Handling currencies

Endpoint Component: CurrencyController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /currency/{systemId}

Endpoint Request Objects: Currency

Endpoint Response Objects: FcuCurrencyResult

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId***- The unique identifier of the FAM instance (AP number)

Request data structure

```
{  
  "currencyCode": "EUR",  
  "conversionValue": "350",  
  "displayPrecision": 2,  
  "isNative": false,  
  "symbol": "Ft"  
}
```

Response Data Structure (Upon Successful Execution)

```
{  
  "resultCode": "SUCCESS",  
  "resultDesc": null  
}
```

The system first performs general request validations ([General request validations and response messages](#)), then carries out endpoint-specific checks:

Validation/Description	Result Code	J	Action Required
The addition or modification of the currency was unsuccessful.	CANNOT_SAVE_CURRENCY	T	Provide a valid currency identifier or data structure.

The native currency (HUF) cannot be deleted.

12.7 Payment methods

The cash register balance and its changes are tracked by the FAM through the management of payment methods. By default, a new FAM instance is configured with three payment methods:

- Hungarian Forint cash
- Hungarian Forint bank card
- Hungarian Forint AFR



12.7.1 Key object descriptions and data fields

12.7.1.1 paymentMethod – Payment method

- **paymentMethod** - Payment method object
 - **id** - Internal unique identifier of the payment method
 - **systemId** - The unique identifier of the FAM instance (AP number)
 - **displayName** - The name of the payment method
 - **moneyCat** - The primary category of the payment method
The valid values are found in the [moneyCat/](#) section.
 - **moneySubCat** - The specific subtype required if the moneyCat is OTHER.
Possible values are described in [moneyCat/](#) subsection.
 - **currency** - The currency of the payment method if it differs from the local currency
 - **sortKey** - The designated position in the display order for the cash register

12.7.2 Querying a payment method

API endpoint group details: FAM interface/PaymentMethod Managing payment methods

Endpoint Component: PaymentMethodController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /payment-method/{systemId}/{paymentMethodId}

Endpoint Response Objects: GetPaymentMethodResponse

Request data structure

For the GET method, no request body is required.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** - The unique identifier of the FAM instance (AP number)
- **paymentMethodId** - Internal unique identifier of the payment method

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "paymentMethod": {  
        "id": 2,  
        "systemId": "C00000001",  
        "displayName": "Credit Card",  
        "moneyCat": "CARD",  
        "moneySubCat": null,  
        "currency": "HUF",  
        "sortKey": "0002"  
    }  
}
```

Data structure field explanation:

- **paymentMethod** – Payment method object
The related data structure description can be found under the [paymentMethod](#) section.



The system first performs general request validations ([General request validations and response messages](#)), then carries out endpoint-specific checks:

Validation/Description	Result Code	J	Action Required
Unknown payment method	CANNOT_GET_PAYMENT_METHOD	T	Provide the correct payment method identifier.

12.7.3 Retrieving all payment methods

API endpoint group details: FAM interface/PaymentMethod – Managing payment methods

Endpoint Component: PaymentMethodController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /payment-method/{systemId}

Endpoint Response Objects: GetAllPaymentMethodResponse

Request data structure

For the GET method, no request body is required.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** - The unique identifier of the FAM instance (AP number)

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "paymentMethods": [  
        {  
            "id": 1,  
            "systemId": "C00000001",  
            "displayName": "Cash",  
            "moneyCat": "CASH",  
            "moneySubCat": null,  
            "currency": "HUF",  
            "sortKey": "0001"  
        },  
        {  
            "id": 2,  
            "systemId": "C00000001",  
            "displayName": "Credit Card",  
            "moneyCat": "CARD",  
            "moneySubCat": null,  
            "currency": "HUF",  
            "sortKey": "0002"  
        },  
        {  
            "id": 3,  
            "systemId": "C00000001",  
            "displayName": "AFR",  
            "moneyCat": "AFR",  
            "moneySubCat": null,  
            "currency": "HUF",  
            "sortKey": "0003"  
        }  
    ]  
}
```



```
        }  
    ]  
}
```

Data structure field explanation:

- **paymentMethods** – Payment method object

The related data structure description can be found under the [paymentMethod](#) section.

12.7.4 Deleting a payment method

API endpoint group details: FAM interface/PaymentMethod – Managing payment methods

Endpoint Component: PaymentMethodController

HTTP Method: DELETE

Context Root: /fam/v1

Endpoint URL: /payment-method/{systemId}/{paymentMethodId}

Endpoint Response Objects: FcuPaymentMethodResult

Request data structure

Not applicable in the case of the DELETE method.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** - The unique identifier of the FAM instance (AP number)
- **paymentMethodId** - The unique identifier of the payment method

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null  
}
```

The system first performs general request validations ([General request validations and response messages](#)), then carries out endpoint-specific checks:

Validation/Description	Result Code	J	Action Required
Payment method cannot be deleted.	CANNOT_DELETE_PAYMENT_METHOD	T	Provide a valid payment method identifier.

12.7.5 Adding or modifying a payment method

API endpoint group details: FCU interface/PaymentMethod – Managing payment methods

Endpoint Component: PaymentMethodController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /payment-method

Endpoint Request Objects: PaymentMethod

Endpoint Response Objects: FcuPaymentMethodResult



Request data structure

```
{  
    "id": "45678",  
    "systemId": "C00000001",  
    "displayName": "SZÉP kártya",  
    "moneyCat": " OTHER_CHANGE_NON_RETURNABLE",  
    "moneySubCat": "SZEP",  
    "currency": "HUF",  
    "sortKey": "0003"  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

The related data structure is the same as the [paymentMethod](#).

- **id** – The internal unique identifier of the payment method; if provided, it edits the payment method; if null, it adds the payment method
- **systemId*** – The unique identifier of the FAM instance (AP number)
- **displayName*** – The name of the payment method
- **moneyCat*** - The primary type of the payment method
- **moneySubCat** - Required for the OTHER moneyCat, specifying a unique subtype.
- **currency*** – The currency of the payment method if different from the local currency
- **sortKey*** – The designated position in the display order for the cash register

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null  
}
```

The system first performs general request validations ([General request validations and response messages](#)), then carries out endpoint-specific checks:

Validation/Description	Result Code	J	Action Required
The addition/modification of the payment method was unsuccessful	CANNOT_SAVE_PAYMENT_METHOD	T	Provide the correct payment method identifier or data structure.

12.7.6 Querying all predefined payment methods

All payment methods defined by the Cash Register Regulation can be queried from the FAM. These serve as templates and are not added to the FAM instance by default. The list is intended to facilitate their addition.

A user interface page can be designed where the user selects the available payment methods they wish to use, and the application can then add these as new payment methods assigned to the FAM instance.

API endpoint group details: FAM interface/PaymentMethod – – Managing payment methods

Endpoint Component: PaymentMethodController

HTTP Method: GET

Context Root: /fam/v1



Endpoint URL: /payment-method/predefined

Endpoint Response Objects: GetAllPaymentMethodResponse

Request data structure

For the GET method, no request body is required.

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "paymentMethods": [  
        {  
            "systemId": null,  
            "displayName": "Cash",  
            "moneyCat": "CASH",  
            "moneySubCat": null,  
            "currency": "HUF",  
            "sortKey": "1",  
            "id": null  
        },  
        {  
            "systemId": null,  
            "displayName": "Credit Card",  
            "moneyCat": "CARD",  
            "moneySubCat": null,  
            "currency": "HUF",  
            "sortKey": "2",  
            "id": null  
        },  
        {  
            "systemId": null,  
            "displayName": "Azonnali fizetés",  
            "moneyCat": "AFR",  
            "moneySubCat": null,  
            "currency": "HUF",  
            "sortKey": "3",  
            "id": null  
        },  
        {  
            "systemId": null,  
            "displayName": "Széchenyi Card",  
            "moneyCat": "SZEP",  
            "moneySubCat": null,  
            "currency": "HUF",  
            "sortKey": "4",  
            "id": null  
        },  
        {  
            "systemId": null,  
            "displayName": "Gift Card",  
            "moneyCat": "OTHER_CHANGE_NON_RETURNABLE",  
            "moneySubCat": "AJÁND",  
            "currency": null,  
            "sortKey": "5",  
            "id": null  
        },  
        {  
    ]  
}
```



```
"systemId": null,  
"displayName": "Loyalty Card",  
"moneyCat": "OTHER_CHANGE_NON_RETURNABLE",  
"moneySubCat": "HÚSÉG",  
"currency": null,  
"sortKey": "6",  
"id": null  
},  
{  
    "systemId": null,  
    "displayName": "Smart Card",  
    "moneyCat": "OTHER_CHANGE_NON_RETURNABLE",  
    "moneySubCat": "SMART",  
    "currency": null,  
    "sortKey": "7",  
    "id": null  
},  
{  
    "systemId": null,  
    "displayName": "Coupon",  
    "moneyCat": "OTHER_CHANGE_NON_RETURNABLE",  
    "moneySubCat": "KUPON",  
    "currency": null,  
    "sortKey": "8",  
    "id": null  
},  
{  
    "systemId": null,  
    "displayName": "Egyéb nem visszajáró",  
    "moneyCat": "OTHER_CHANGE_NON_RETURNABLE",  
    "moneySubCat": "EGYEB-*",  
    "currency": null,  
    "sortKey": "9",  
    "id": null  
},  
{  
    "systemId": null,  
    "displayName": "Göngyölegjegy",  
    "moneyCat": "OTHER_CHANGE_RETURNABLE",  
    "moneySubCat": "GÖNGY",  
    "currency": "HUF",  
    "sortKey": "10",  
    "id": null  
},  
{  
    "systemId": null,  
    "displayName": "Egyéb visszajáró",  
    "moneyCat": "OTHER_CHANGE_RETURNABLE",  
    "moneySubCat": "EGYEB-*",  
    "currency": null,  
    "sortKey": "11",  
    "id": null  
}  
]  
}
```



Data structure field explanation:

- **paymentMethods** – List of payment method objects

The description of the data structure elements in the list can be found under the [paymentMethod](#) section.

12.8 Peripheral management

The printing and print control of receipts and receipt copies are carried out by the e-cash register client application in the manner described below.

At each step of receipt creation, the FAM compiles the data to be printed or displayed—including calculated values such as the total amount, rounding, and the total amount converted to another currency—using the numerical representation (decimal comma or point, thousand separators) according to the given language and currency settings and returns it to the fiscal application.

The e-cash register client application generates the print image exclusively from the data received from the FAM, applying control codes appropriate to the configured printer and display type (e.g., bold or double-height text, paper cutting) and column width (number of characters per line). The e-cash register client application prints the receipt or receipt copy.

12.8.1 Peripheral settings

The FAM provides the ability to store and retrieve peripheral settings (printer, customer display). This is an optional feature for cloud-based e-cash register clients, allowing certain peripheral management settings to be stored outside the client application.

12.8.2 Querying settings

API endpoint group details: FAM interface/Telemetry – System functions

Endpoint Component: SystemController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /system/peripheral-settings/{systemId}?key=...

Endpoint Response Objects: PeripheralSettingsResponse

Request data structure

For the GET method, no request body is required.

The URL parameter from the request:

* Fields marked with * are mandatory

- **systemId*** - The unique identifier of the FAM instance (AP number)

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "settings": {  
        "key": "value"  
    }  
}
```



Data structure field explanation:

- **settings** – The retrieved value is returned in **key-value** pairs

The system first performs general request validations ([General request validations and response messages](#)), then carries out endpoint-specific checks:

Validation/Description	Result Code	J	Action Required
The retrieval of peripheral settings has failed.	FAILED_TO_GET_PERIPHERAL_SETTINGS	T	The format of the key provided in the request is incorrect

12.8.3 Storing a setting

API endpoint group details: FAM interface/Telemetry – System functions

Endpoint Component: SystemController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /system/peripheral-settings

Endpoint Request Objects: PeripheralSettingsRequest

Endpoint Response Objects: FcuSystemResult

Request data structure

```
{  
    "systemId": "C12345678",  
    "settings": {  
        "key1": "value1",  
        "key2": "value2",  
        ...  
    }  
}
```

Data structure field explanation:

- **systemId*** - The unique identifier of the FAM instance (AP number)
- **settings*** - The submitted key-value pairs

Example of setting printing parameters (57 mm paper width, 203 dpi resolution, 384 pixels width, 48 characters per line printing):

```
{  
    "systemId": "C12345678",  
    "settings": {  
        "sys.printer.selected": "client default",  
        "sys.client.printer.default.dpi": "203",  
        "sys.client.printer.default.imageWidthPx": "384",  
        "sys.client.printer.default.paperWidthChr": "48"  
    }  
}
```

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null  
}
```



The system first performs general request validations ([General request validations and response messages](#)), then carries out endpoint-specific checks:

Validation/Description	Result Code	J	Action Required
The saving/modification of peripheral settings has failed	FAILED_TO_SET_PERIPHERAL_SETTINGS	T	Provide the correct peripheral setting identifier or data structure.

12.9 Receipt management

The handling of fiscal receipts and data provision to the tax authority is ensured by FAM. Currently, FAM supports the creation of ten types of receipts. These receipts can be categorized into three groups:

- Sales Receipts
- Report Receipts
- Other Receipts

The distinction arises not only from their function but also from their method of creation. Printing of receipts is the responsibility of the client application. The printout always contains the data returned by FAM for the given receipt.

Sales Receipts

Sales receipts, as the name suggests, are used for performing sales-related tasks.

Their creation involves multiple steps, as follows:

- Creation and item addition
- Adding additional items (optional, this step can be repeated multiple times)
- Closing

These steps can be passed to FAM in two or more consecutive API calls.

The available sales receipts are:

- Receipt
- Simplified Invoice
- Cancellation Receipt
- Modification Receipt
- Invoice

Report receipts

Report receipts are used for operations related to the fiscal day and provide information about the current state of the fiscal day(s). Their creation is done in a single step via a single API endpoint call.

The available report receipts are:

- Cash Register Opening Receipt
- Cash Register Report Receipt
- Daily Sales Report
- Daily Receipt List Report



Other receipts

The Cash Register Regulation allows for the creation of additional documents when necessary for cash register operation, business processes, or other reasons. This category includes the Cash Movement Receipt, which allows for modifying the cash register's content.

- Cash Movement Receipt
 - Used to log cash deposits and withdrawals and facilitate currency exchange.
 - Created similarly to report receipts through a single API endpoint call.
- Custom Receipt
 - Custom receipts cannot modify the cash register balance.
 - Their creation follows the same method as sales receipts, in two or more steps (creation and adding informational items, optionally adding more informational items, and closing).

12.9.1 Automatic end-of-day closure

The fiscal day must conclude on the current calendar day. The cash register operator is required to close the fiscal day before 23:55. If this is not done, FAM will automatically execute the end-of-day closure starting from 23:55.

Between 23:55 and 00:00, no fiscal operations can be performed. Any open receipts will be interrupted, and the cash register report and daily sales report receipts will be automatically generated. In the case of automatic end-of-day closure, the new fiscal day can only be opened starting from 00:00.

12.9.2 Response messages and error codes

Beyond general validation, the document interface also performs checks specific to the endpoint group. The table below summarizes the general validation response messages applicable to this endpoint group.

If a validation-related field is not relevant for a particular endpoint, the validation will not be performed. Descriptions of specific validations related to individual endpoints can be found in their respective sections.

Validation/Description	Result Code	J	Action Required	T+M
Automatic end-of-day closure is in progress	FISCAL_DAY_NEEDS_TO_BE_CLOSED	T	Wait for the process to be completed, which can be monitored using the FAM status query	ÉJE R
The client has initiated an operation that can only be executed with an open fiscal day. Except for the cash register opening, all document-related calls fall into this category	FISCAL_DAY_CLOSE_D	P	The fiscal day must be opened by generating a cash register opening receipt.	ÉJE R
The client attempted to create a new receipt, but a previously created receipt has not been closed (not printed on paper).	FISCAL_DAY_HAS_OPENED_DOCUMENT	P	The open document ID can be obtained via the FAM instance status query. Its content can then be queried to ensure proper closure or cancellation	ÉJE R
The referenced receipt ID does not exist or has been incorrectly provided.	DOCUMENT_CANNOT_BE_FOUND	P	Verify that the receipt data provided is correct.	ÉJE RTL



Validation/Description	Result Code	J	Action Required	T+M
The item category (itemCat) is invalid	INVALID_ITEM_CATEGORY	P	Provide a valid itemCat in the request.	ÉT
A subcategory has been specified in an invalid case for payment methods	INVALID_SUB_CATEGORY	P	The moneySubCat field should only contain a value if the "moneyCat" value starts with "OTHER..". The correct category must be provided in the moneyCat field.	ÉT
An unknown currency has been specified for the FAM instance.	INVALID_CURRENCY	T	Add the selected currency to the FAM instance (see "Currency Management" section).	ÉL
The sum of the specified payment methods does not match the total amount of the receipt.	PAYMENT_MISCALCULATED_TOTAL	T	Ensure that the correct payment amounts are specified. If this error occurs, the "remainingSum" field in the response object will indicate the missing amount	ÉL
There is insufficient cash in the drawer to complete the requested transaction (less cash in the drawer than the change required)	NOT_ENOUGH MONEY_IN_DRAWER	T	Adjust the payment methods, cancel the receipt, deposit change using a cash movement receipt, then retry the receipt creation.	ÉJRL
The total amount of the receipt exceeds the legal limit allowed by the VAT law.	INVALID_DOCUMENT_TOTAL	T	Modify or close the receipt without the item(s) causing the overflow.	ÉTL
Receipt data has been successfully recorded, but FAM could not generate the output QR code. The document is closed.	SUCCESS_WITH_NO_QR	P	The receipt is closed, but FAM is unable to return the output QR code to the client. If the user later requests to display or print a receipt copy, FAM may be able to provide it retroactively.	ÉJERL
Receipt data recording failed due to a server-side error, leaving it open in FAM.	DOCUMENT_CONVERSATION_ERROR	T	Attempt to cancel the receipt. If unsuccessful, wait for the server-side error to be resolved.	ÉJERL
Receipt data has been successfully recorded, but FAM could not upload the receipt to the fiscal archive. The document is closed	SUCCESS_WITH_NO_UPLOAD	P	Do not play the success confirmation image or sound. Fiscal operations may continue. FAM will automatically retry the upload in the background until it succeeds.	ÉJERL
The receipt ID provided in the request is invalid.	INVALID_DOCUMENT_ID	T	Provide a valid receipt ID in the request.	ÉETL
The payment data includes a ROUND/CHANGE payment category.	INVALID_PAYMENT_CATEGORY	T	Remove the ROUND/CHANGE payment category from the payment data.	ÉL
Among the payment data, there is an amount of money with an opposite sign.	INVALID_PAYMENT_VALUE	T	Removal of data structure with incorrect sign from the payment data	ÉL
In the data structure of the receipt closing HTTP request, a service charge is included, however its use may not be possible depending on the FAM configuration setting.	FORBIDDEN_USE_OF_SERVICE_FEE	P	The value of the serviceFee field must be set to null.	ÉL



T+M – Specifies which document type and which document processing operation the given validation applies to:

- Type:
 - É – Sales document
 - J – Report document
 - E – Other document
- Operation:
 - R – Recording/Creation
 - T – Adding an item
 - L – Closing

Specific response messages for individual endpoints—if any—are indicated at the respective endpoint.

12.9.3 Key object descriptors and data fields

12.9.3.1 DocumentData - Receipt Image Data

Each receipt endpoint **response** may contain receipt image-related data. The data is contained in a list of type **DocumentData**.

The receipt image data structure is divided into four main groups:

- **Receipt header (docCreate)**
 - Elements of the receipt header
 - Fiscal day serial number
 - Receipt identifier
 - Receipt title
 - Seller company details
 - Seller company name
 - Seller company headquarters
 - Seller company tax number
 - Store details
 - Store name
 - Store headquarters
 - Three-letter currency code
 - Short currency name
 - Receipt-specific data
- **Receipt Body:**
 - Sales item (receiptItem)
 - Report item (reportItem)
 - Custom item (customInfo)
- **Receipt footer**
 - Elements of the receipt footer (docClose)
 - AP number (systemId)
 - Footer remarks
 - Receipt creation timestamp
 - NAV verification code
 - Receipt number



- Indicator flag for receipt interruption
- Sales receipt specific data

The **subclasses of the DocumentData** class can include the following:

- DocCreate
- ReceiptItem
- ReportItem
- CustomInfo
- DocClose

12.9.3.1.1 Structure of DocCreate

```
{  
    "@type": "docCreate",  
    "fiscalDayNo": "1",  
    "docId": "1",  
    "docTitle": "E-NYUGTA COPY",  
    "headNote": "",  
    "operatorSite": {  
        "effectiveDate": 0,  
        "shop": {  
            "address": {  
                "addressType": "DETAILED",  
                "countryCode": "HU",  
                "postCode": "9512",  
                "city": "Ostffyasszonyfa",  
                "streetName": "Szarvas",  
                "publicPlaceCategory": "utca",  
                "houseNumber": "32"  
            },  
            "shopName": "Szuper Bolt Kft",  
            "shopShortName": "SzB Kft."  
        },  
        "taxpayer": {  
            "address": {  
                "addressType": "DETAILED",  
                "countryCode": "HU",  
                "postCode": "9512",  
                "city": "Ostffyasszonyfa",  
                "streetName": "Szarvas",  
                "publicPlaceCategory": "utca",  
                "houseNumber": "32"  
            },  
            "taxpayerName": "Béla",  
            "taxpayerShortName": "Béla",  
            "taxNumber": {  
                "taxpayerId": "30000003",  
                "vatCode": "3",  
                "countyCode": "33"  
            }  
        },  
        "temporalType": "CURRENT"  
    },  
    "temporalType": "CURRENT"  
}
```



```
},
"docCreationDate" : "2024.07.10 15:11:41",
"moneyCode" : "HUF",
"moneyShortName" : "Ft",
"logo" : null
"itemHeader" : "GYŰJTŐ MEGNEVEZÉS\nMENNYISÉG * EGYSÉGÁR\tÉRTÉK",
"taxNumberHeader" : "ADÓSZÁM:",
{returnReason" : null,
"voidReason" : null,
"sourceDocNo" : null,
"invoiceNo" : null,
"fulfillmentDate" : null,
"paymentDue" : null,
"paymentType" : null,
"billToHeader" : null,
"sourceDocType" : null
}
```

Data structure field explanation:

- **@type** – DocumentData subcategory (=docCreate)
- **fiscalDayNo** - Serial number of the fiscal day
- **docId** - Unique identifier of the receipt within the FAM system.
- **docTitle** – Title of the receipt
- **headNote** – Address note
- **operatorSite** – Operator and Location Data
 - **effectiveDate** – Start date of data validity
 - **shop** – Location data
 - address – Store location
The related data structure description is found under the [address](#) section.
 - shopName – Name of the store
 - shopShortName – Short name of the store
 - **taxpayer** – Operator data
The related base data structure description is found under the [taxpayer object](#) section. In this response, the taxpayer data is supplemented with the [address](#) data in addition to what is described under "taxpayer".
 - **temporalType** – Type of data validity (=CURRENT)
The value set can be found under the section [\(TaxRates\)/type / temporalType](#).
- **docCreationDate** – Timestamp of when the receipt was created
- **moneyCode** – Currency of the receipt
- **moneyShortName** - Short name of the receipt currency
- **logo** – Branding logo, currently not in use; ignored by FAM
- **itemHeader** – Header of items on the receipt image (the data structure includes it only in the case of sales receipts)
- **taxNumberHeader** – Header of the tax number on the receipt image
- **returnReason** – Reason for modification (the data structure includes it only in the case of a Modification receipt)



- **voidReason** – Reason for cancellation (the data structure includes it only in the case of a Cancellation receipt)
- **sourceDocNo** – Serial number of the referenced receipt (the data structure includes it only in the case of Modification and Cancellation receipts)
 - **text** – text
 - **value** – value
- **invoiceNo** – Invoice serial number (the data structure includes it only in the case of Invoice and Simplified invoice receipts)
 - **text** – text
 - **value** – value
- **fulfillmentDate** – Date of performance (the data structure includes it only in the case of Invoice and Simplified invoice receipts)
 - **text** – text
 - **value** – value
- **paymentDue** – Payment deadline (the data structure includes it only in the case of Invoice receipts)
 - **text** – text
 - **value** – value
- **paymentType** – Method of payment (the data structure includes it only in the case of Invoice receipts)
 - **text** – text
 - **value** – value
- **billtoHeader** – Header of the customer on the receipt image
- **sourceDocType** – Type of the referenced receipt (the data structure includes it only in the case of Modification and Cancellation receipts)
 - RECEIPT – Receipt
 - INVOICE – Invoice
 - SIMPLE_INVOICE – Simplified invoice

12.9.3.1.2 Structure of ReceiptItem

```
{  
    "@type": "receiptItem",  
    "itemRef": null,  
    "itemId" : 1,  
    "itemName" : "Teszt item 1",  
    "itemArticleNo" : "Z3344505",  
    "itemUnitPrice" : "540,05",  
    "itemTax" : null,  
    "itemQty" : "1",  
    "itemUnit" : "PIECE",  
    "itemCat" : "SALE",  
    "itemDept" : "A",  
    "itemSum" : "540",  
    "itemSumNet" : null,  
    "itemCustomInfo" : [ {  
        "@type" : "text",  
        "key": "A00004_BASIC",  
        "description": "Leírás",  
        "orderId" : 1,  
        "text" : "This is a basic text info",  
    } ]  
}
```



```
    "alignment" : "CENTER"
  },
  {
    "@type" : "image",
    "key": "N00005_LOGO",
    "description": "Leírás",
    "orderId" : 1,
    "imageId" : "item_2"
  },
  {
    "@type" : "barcodeQr",
    "key": "N00004_SERIAL",
    "description": "Leírás",
    "orderId" : 2,
    "data" : "1231231231",
    "ec" : 1,
    "size" : 3,
    "mode" : 3
  }
]
```

Data structure field explanation:

- The fields are identical to those described in the [receiptItems](#) section.

12.9.3.1.3 Structure of ReportItem

```
{
  "@type": "reportItem",
  "textType": null,
  "label" : "BEFIZETÉS (HUF)",
  "value" : "4 000"
}
```

Data structure field explanation:

- **@type** – DocumentData subcategory
- **textType** - Text formatting enum, possible values: *TITLE*, *HEADER*, *FOOTER*, *SEPARATOR*
- **label** – label
- **value** - value

12.9.3.1.4 Structure of CustomInfo (item) –Custom Information

For unique informational items, all fields must be provided for the given type.

Textual information

```
{
  "@type": "text",
  "key": "A00004_MEGJ",
  "description": "Leírás",
  "textType": null,
  "text": "comment",
  "alignment": "CENTER",
  "orderId": 1
}
```



Data structure field explanation:

- **@type** - Informs the FAM Document interface about the type of data structure submitted in the HTTP request
- **key** –The name of the supplementary data as described in the chapter “[Optional supplementary fields for documents.](#)”
- **description** –The explanatory description of the supplementary data as described in the chapter “[Optional supplementary fields for documents.](#)”
- **textType** - Text formatting enum, possible values: *TITLE, HEADER, FOOTER, SEPARATOR*
- **text**- The content of the textual information
- **alignment** - The position of the textual information on the printed receipt
 - Possible values *CENTER, LEFT, RIGHT*
- **orderId** - The display order number of the informational item

Important additions:

The parameters for text formatting apply only to the receipt images generated by the FAM, in the NAV data service the text is submitted without formatting.

The text does not need to fit into one line of the printer, it may contain multi-line continuous text, and the “\n” code can be used for line breaks.

If the “key” field contains the “_BIZALMAS” postfix – for example, if a discount card number must be displayed in the field – the FAM will not display the given supplementary text element on the receipt image. If on the printed copy of the receipt we want to confirm the reading of the discount card, this can be done in a separate supplementary text element, with the card number partially masked.

Picture{

```
    "@type": "image",
    "key": "N00005_KEP",
    "description": "Leírás",
    "imageId": "string",
    "orderId": 2
}
```

Data structure field explanation:

- **@type**- Informs the FAM Document interface about the type of data structure sent in the HTTP request
- **key** –The name of the supplementary data as described in the chapter “[Optional supplementary fields for documents.](#)”
- **description** –The explanatory description of the supplementary data as described in the chapter “[Optional supplementary fields for documents.](#)”
- **imageId**- The identifier of the image in the FAM file storage
- **orderId**- The display order number of the informational item

Barcode 1D

```
{
    "@type": "barcode1D",
    "key": "N00004_CODE39_KOD",
    "description": "Leírás",
    "orderId": 3,
    "barcodeType": "CODE39",
    "data": "123123123123",
```



```
{  
    "width": 20,  
    "height": 20  
}
```

Data structure field explanation:

- **@type**- Informs the FAM Document interface about the type of data structure sent in the HTTP request
- **key** –The name of the supplementary data as described in the chapter “[Optional supplementary fields for documents](#).”
- **description** –The explanatory description of the supplementary data as described in the chapter “[Optional supplementary fields for documents](#).”
- **orderId**- The display order number of the informational item
- **barcodeType**– Barcode type
 - Possible values: *UPC_A, UPC_E, EAN13, EAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1_128, GS1_DB_O, GS1_DB_T, GS1_DB_L, GS1_DB_E*
- **data**- The content of the barcode
- **width**- The width of a single column of the barcode
- **height**- The height of the barcode

Barcode DataMatrix

```
{  
    "@type": "barcodeDataMatrix",  
    "key": "N00004_DATA_MATRIX",  
    "description": "Leírás",  
    "orderId": 4,  
    "data": "123123123123",  
    "columns": 10,  
    "rows": 10,  
    "size": 20,  
    "mode": 1  
}
```

Data structure field explanation:

- **@type**- Informs the FAM Document interface about the type of data structure sent in the HTTP request
- **key** –The name of the supplementary data as described in the chapter [Optional supplementary fields for documents](#).”
- **description** –The explanatory description of the supplementary data as described in the chapter [Optional supplementary fields for documents](#).”
- **orderId**- The display order number of the informational item
- **data**- The content of the barcode
- **columns**- Number of the columns in the barcode
- **rows**- The number of rows in the barcode
- **size**- The size of the barcode
- **mode** - The encoding mode of the barcode



Barcode PDF417

```
{  
    "@type": "barcodePdf417",  
    "key": "N00004_EZ_EGY_PDF417",  
    "description": "Leírás",  
    "orderId": 5,  
    "data": "123123123123",  
    "columns": 10,  
    "rows": 10,  
    "width": 20,  
    "rowHeight": 20,  
    "ec": 2,  
    "mode": 1  
}
```

Data structure field explanation:

- **@type**- Informs the FAM Document interface about the type of data structure sent in the HTTP request
- **key** –The name of the supplementary data as described in the chapter “[Optional supplementary fields for documents.](#)”
- **description** –The explanatory description of the supplementary data as described in the chapter “[Optional supplementary fields for documents.](#)”
- **orderId**- The display order number of the informational item
- **data**- The content of the barcode
- **columns**- Number of the columns in the barcode
- **rows**- The number of rows in the barcode
- **width**- The width of a single column of the barcode
- **rowHeight**- The heights of the barcode row
- **ec** - The error correction mode of the barcode
- **mode** - The encoding mode of the barcode

QR code

```
{  
    "@type": "barcodeQr",  
    "key": "N00004_QR_KOD",  
    "description": "Leírás",  
    "orderId": 6,  
    "data": "123123123123",  
    "ec": 1,  
    "size": 10,  
    "mode": 4  
}
```

Data structure field explanation:

- **@type**- Informs the FAM Document interface about the type of data structure sent in the HTTP request
- **key** –The name of the supplementary data as described in the chapter “[Optional supplementary fields for documents.](#)”
- **description** –The explanatory description of the supplementary data as described in the chapter “[Optional supplementary fields for documents.](#)”
- **orderId**- The display order number of the informational item



- **data** - The content of the barcode
- **ec** - QR code error correction mode, represented as an integer:
 - **0**: „M” (Medium)
 - **1**: „L” (Low)
 - **2**: „H” (High)
 - **3**: „Q” (Quartile)
- **size** - The QR code size, the "zoom factor"; if the value is 1, then a QR code dot size is 1x1 pixel, if 2, then 2x2 pixels, etc.
- **mode** - A QR kód kódolási módja
 - **4**: The QR code encoding mode o 4: The FAM currently supports only the value 4 ("byte encoding")

In the case of line and dot codes, the data in the “key” field can either start with “N00003_GYORSKOD” or “N00004_”. If the FAM receives one of these types with a different “key” value, it automatically prefixes it with “N00004_”.

For any supplementary element, the content of the “description” field is displayed above the given element on the receipt image. Since the field cannot be empty, if there is no need to display the “description,” a single non-breaking space (U+00A0) or zero-width space (U+200B) character must be entered into the field.

12.9.3.1.5 Structure of DocClose

```
{  
    "@type": "docClose",  
    "systemId" : "Y19500027",  
    "docCloseDate" : {  
        "text" : "DÁTUM:",  
        "value" : "2025.03.18. 13:41:30"  
    },  
    "docValidationCode" : {  
        "text" : "NAV ELLENŐRZŐ KÓD",  
        "value" : "52Y07"  
    },  
    "docNo" : {  
        "text" : "BIZONYLATSZÁM",  
        "value" : "NY-Y19500027/30000003/0001/00001"  
    },  
    "docInterruption" : false,  
    "docTotal" : {  
        "text" : "ÖSSZESEN",  
        "value" : "13 996"  
    },  
    "docTotalNet" : null,  
    "docTotalTax" : null,  
    "serviceFee" : {  
        "text" : "FELSZOLGÁLÓI DÍJ",  
        "value" : "100"  
    },  
    "docCustomInfo" : [ {  
        "@type" : "text",  
        "key": "D00011_BASIC_TEXT",  
        "description": "Leírás",  
    }],  
}
```



```
"orderId" : 1,
"text" : "This is a basic text info",
"alignment" : "CENTER"
}, {
"@type" : "image",
"key": "N00005_LOGO",
"description": "Leírás",
"orderId" : 1,
"imageId" : "1"
}, {
"@type" : "barcode1D",
"key": "N00003_GYORSKOD",
"description": "Kapukód",
"orderId" : 2,
"barcodeType" : "CODE128",
"data" : "123123123123",
"width" : 100,
"height" : 100
}, {
"@type" : "barcodeDataMatrix",
"key": "N00004_DATA",
"description": "Leírás",
"orderId" : 3,
"data" : "123123123123",
"columns" : 3,
"rows" : 3,
"size" : 100,
"mode" : 100
}, {
"@type" : "barcodePdf417",
"key": "N00004_PDF417_KOD",
"description": "Leírás",
"orderId" : 4,
"data" : "123123123312",
"columns" : 3,
"rows" : 3,
"width" : 100,
"rowHeight" : 50,
"ec" : 3,
"mode" : 3
}, {
"@type" : "barcodeQr",
"key": "N00004_EGYEDI_AZON",
"description": "Leírás",
"orderId" : 5,
"data" : "1231231231",
"ec" : 1,
"size" : 3,
"mode" : 3
} ],
"paymentDetails" : [ {
"name" : "BANKKÁRTYA",
"moneyCat" : "CARD",
"moneyAmount" : "13 996",
"moneyLocalValue" : "13 996",
```



```
"currency" : "HUF",
"currencyXchRate" : "1",
"currencySymbol" : "Ft",
"isLocalCurrency" : true
}, {
"name" : "VISSZAJÁRÓ",
"moneyCat" : "CHANGE",
"moneyAmount" : "0",
"moneyLocalValue" : "0",
"currency" : "HUF",
"currencyXchRate" : "1",
"currencySymbol" : "Ft",
"isLocalCurrency" : true
} ],
"taxHeader" : null,
"taxList" : null
```

Data structure field explanation:

- **@type** – DocumentData subcategory (= docClose)
- **systemId** - The unique identifier of the FAM instance (AP number)
- **docCloseDate** – Date of receipt closure
 - **text** – title, text
 - **value** - value
- **docValidationCode** – NAV verification code
 - **text** – title
 - **value** - value
- **docNumber** – Receipt serial number
 - **text** – title
 - **value** - value
- **docInterruption** – Field indicating receipt interruption
- **docTotal** – Receipt total amount (this data structure is only included for sales receipts)
 - **text** – title
 - **value** - value
- **docTotalNet** – Invoice net total amount (the data structure includes it only in the case of invoices and receipts cancelling/modifying them)
 - **text** – title
 - **value** – value
- **docTotalTax** - Invoice VAT amount (the data structure includes it only in the case of invoices and receipts cancelling/modifying them)
 - **text** – title
 - **value** – value
- **serviceFee** - Service charge (the data structure includes it only in the case of sales receipts)
 - **text** – title
 - **value** – value
- **docCustomInfo** – Custom information related to the receipt. The CustomInfo subclasses are explained in the “[Receipt Management](#)” section under “[Adding an item](#)” subsection.



- **paymentDetails** – Payment details, the fields of the paymentDetails structure are described later in this section (this data structure is only included for sales receipts)
- **taxHeader** - VAT rate breakdown header (the data structure includes it only in the case of Invoice)
- **taxList** – Breakdown of invoice item amounts by VAT rate (the data structure includes it only in the case of Invoice)
 - **percent** – VAT rate percentage value
 - **net** – total net value of items
 - **gross** – total gross value of items
 - **tax** – total VAT value of items

12.9.3.2 Attachment – Receipt attachment

Sales and other receipts may contain an attachment with arbitrary data content, which is placed exclusively in the customer envelope, so only the customer has access to its content. With the help of the data structure described here, warranty information and coupons specified in the [Use Cases](#) subsection can be attached to the receipt.

The attachment can contain two types of data:

- **Text data**, whose fields correspond to the additionalData object fields used in the NAV-I interface XSD:
 - **key** – The unique identifier of the data field, up to 255 characters, following the pattern “[A-Z][0-9]{5}[][_][A-Z0-9]{1,249}”.
 - **description** – Textual description of the content of the data field, up to 255 characters.
 - **value** – The value of the data, up to 512 characters.
- **File**, which may be up to 512 kB (524288 bytes) in size with base64 encoding. There are no content restrictions, multiple source files can also be included in a ZIP archive (within the size limit). The file object fields also match those of the NAV-I XSD AttachmentType type:
- **fileExtension** – the file extension, which helps with display, e.g. “pdf”, “docx”, “zip” etc., provided in up to 10 characters
 - **fileBinary** – the file content with base64 encoding.

The text data may belong to the entire receipt or only to the receipt item specified in the text data. The file can only be attached to the entire receipt, but the text data at the item level can reference the file or, for example, a specific filename within the attached ZIP archive.

The structure of the FAM attachment data is shown in the following example:

{

```
"attachmentHeads": [  
  {  
    "key": "",  
    "description": "",  
    "value": ""  
  }  
,  
  "attachmentItems": [
```



```
{
  "itemId": 1,
  "key": "",
  "description": "",
  "value": ""
}
],
"attachmentFile": {
  "fileExtension": "",
  "fileBinary": ""
}
}
```

The attachment can be submitted in the docClose request

Data structure field explanation:

* Fields marked with * are mandatory

- **attachmentHeads** – List of receipt-level text attachments, up to ten (10) text attachments can be added to the receipt.
 - **key*** – data field identifier
 - **description*** – description
 - **value*** - value
- **attachmentItems** - List of item-level text attachments, up to ten (10) text attachments can be added to an item.
 - **itemId*** - the identifier of the receipt item to which the text data belongs
 - **key*** – key, data field identifier
 - **description*** – description
 - **value*** - value
- **attachmentFile** – a file to be attached
 - **fileExtension*** – extension of the file
 - **fileBinary*** – content of the file

12.9.3.3 PaymentDetails - Receipt Monetary Data

For sales receipts, it is generally required that the total amount of the payment methods provided cannot be less than the total amount of the receipt. Payment methods must be submitted to the FAM separately by type, in distinct data structures. If the specified amount is insufficient to settle the receipt, the FAM returns an error message, and the receipt closure does not occur.

For Void Receipts, Modification Receipts, and Cash Movement Receipts, the payment amounts must be submitted **with a negative sign** when making a payment.

```
"paymentDetails": [
  {
    "name": "Cash",
    "moneyCat": "CASH",
    "moneySubCat": null,
    "moneyAmount": "5000.00",
    "moneyLocalValue": "5000.00",
    "currency": "HUF",
    "currencyXchRate": "1",
```



```
        "currencySymbol": "Ft",
        "isLocalCurrency": true
    },
    {
        "name": "Change",
        "moneyCat": "CHANGE",
        "moneySubCat": null,
        "moneyAmount": "-3100.00",
        "moneyLocalValue": "-3100.00",
        "currency": "HUF",
        "currencyXchRate": "1",
        "currencySymbol": "Ft",
        "isLocalCurrency": true
    },
    {
        "name": "Round",
        "moneyCat": "ROUND",
        "moneySubCat": null,
        "moneyAmount": "-2.00",
        "moneyLocalValue": "-2.00",
        "currency": "HUF",
        "currencyXchRate": "1",
        "currencySymbol": "Ft",
        "isLocalCurrency": true
    }
],
]
```

Data structure field explanation:

- **paymentDetails** - Payment information (extended with rounding, change, and foreign currency usage data structure)
 - **name** - Name of the payment method
 - **moneyCat*** - Type of payment method
 - **moneySubCat** - Subtype of payment method
 - **moneyAmount** - Amount paid
 - **moneyLocalValue** - Amount in HUF (if payment is made in local currency, this value matches the **moneyAmount**)
 - **currency** - Three-letter currency code of the payment method
 - **currencyXchRate** - Exchange rate of the payment method; for payments in local currency, this value is always 1
 - **currencySymbol** - Symbol of the payment method
 - **isLocalCurrency** - Local currency indicator (for HUF, the value is true)

The possible values for payment method types and subtypes are listed under the [paymentMethod](#) section.

For receipt requests that involve monetary values, the following fields must be filled according to the mandatory structure of the object:

- name
- moneyCat (and moneySubCat if moneyCat has the prefix OTHER)
- moneyAmount
- currency



The remaining fields in the object are automatically calculated, populated, and returned in response messages by the FAM:

- currencyXchRate – Based on the exchange rate set in currency management
- currencySymbol – Derived from FAM currency master data
- isLocalCurrency – Derived from FAM currency master data
- moneyLocalValue – Calculated value: Computed based on the exchange rate (conversionValue) set for the given currency in currency management and the amount paid (moneyAmount) provided in the request.

12.9.3.4 receiptItems - Receipt Item Data

- **receiptItems*** - Data structure for items added to the receipt
 - **itemName*** - Name of the item as it appears on the receipt
 - **itemArticleNo** Article number associated with the item
 - **itemUnitPrice*** - Unit price of the item in HUF as listed on the receipt, in case of invoice, the item's net unit price
 - **itemQty*** - Quantity of the item as listed on the receipt
 - **itemUnit*** Unit of measurement for the item as listed on the receipt
 - **itemCat*** - The nature/category of the given item
 - **itemDept*** - The sales department code as listed on the receipt (regex: [A-E]N|TAM|AAM|EAM|ATK|TRA|SEC|ART|ANT|EUE|HO)
 - **itemCustomInfo** - Custom information related to the item
 - The description of custom information fields that can be specified in the data structure is found under [CustomInfo \(item\)](#) structure.
 - **itemRef** – the sequence number (itemId) of the item in the original receipt.
This field is mandatory for Void or Modification receipts.
 - **itemSubUnit** - The unique unit of measurement for the item as displayed on the receipt.

12.9.3.4.1 itemCat - Item type value set:

- **SALE** - "n": sale
- **VOID_SALE** - "ns": void sale
- **DISCOUNT** - "e": discount
- **VOID_DISCOUNT** - "es": void discount
- **NB_DISCOUNT** - "k": non-business discount
- **VOID_NB_DISCOUNT** - "ks": void non-business discount
- **SURCHARGE** - "f": surcharge
- **VOID_SURCHARGE** - "fs": void surcharge
- **EMPTIES** - "g": returnable packaging
- **VOID_EMPTIES** - "gs": void returnable packaging
- **RETURN** - "v": returned goods
- **VOID_RETURN** - "vs": void returned goods

12.9.3.4.2 itemUnit - Item quantity unit value set:

- **PIECE** – piece
- **KILOGRAM** – kilogram
- **TON** – ton



- **KWH** – kilowatt hour
- **DAY** – day
- **HOUR** – hour
- **MINUTE** – minute
- **MONTH** – month
- **LITER** – liter
- **KILOMETER** – kilometer
- **CUBIC_METER** – cubic meter
- **SQUARE_METER** – square meter
- **LINEAR_METER** – linear meter
- **METER** – meter
- **CARTON** – carton
- **PACK** – pack
- **OWN** – custom unit; in this case, the itemSubUnit field must also be filled.

12.9.3.5 billTo – Customer details

```
{  
    "name": "Teszt Elek",  
    "address": {  
        "addressType": "SIMPLE",  
        "countryCode": "HU",  
        "postCode": "1000",  
        "city": "Budapest",  
        "additionalAddressDetail": "Vas utca 33"  
    },  
    "taxNumber": {  
        "taxpayerId": "30000003",  
        "vatCode": "3",  
        "countyCode": "33"  
    },  
    "groupMemberTaxNumber": {  
        "taxpayerId": "40000004",  
        "vatCode": "4",  
        "countyCode": "44"  
    },  
    "communityTaxNumber": null,  
    "thirdCountryTaxNumber": null,  
    "customerVatStatus": "DOMESTIC",  
    "invoiceType": "ELECTRONIC",  
    "bankAccountNo": "123123123123123123"  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **name*** - Customer's name
- **address*** - Customer's address, the related data structure description can be found in the [address](#) section under addressType = SIMPLE



- **taxNumber*** - Tax number, the related data structure description can be found in the [taxNumber](#) section. If the customer is a member of a VAT group, the VAT group tax number (xxxxxxxxx-5-xx) must be indicated in this field.
- **groupMemberTaxNumber** – Group member tax number (xxxxxxxxx-4-xx). It can only be filled in if the VAT group tax number was provided in the taxNumber field. The description of the related data structure can be found in the taxNumber section.
- **communityTaxNumber** – EU VAT number
- **thirdCountryTaxNumber** – Third-country tax number
- **customerVatStatus** - Customer's VAT status, its value set is detailed in the "[Key Data Fields \(and their value sets\)](#)" section.
- **invoiceType** – Document type, its value set is detailed in the "[Key Data Fields \(and their value sets\)](#)" section.
- **bankAccountNo** – Bank account number

12.9.4 Cash Register opening receipt

The Cash Register Opening Receipt is responsible for opening the fiscal day and contains the initial cash amount in the register, i.e., the amount remaining in the business since the previous day's closing. Once created, it allows for further sales and report receipts to be generated. The currently open fiscal day must be closed on the same calendar day as its opening. If this is not done, the FAM will automatically perform the daily closing at the end of the calendar day (more details can be found in the [Automatic end-of-day closure](#) section).

The creation of the Cash Register Opening Receipt is executed with a single API call.

12.9.4.1 Creating a Cash Register opening receipt

The Cash Register Opening Receipt can only be created if the FAM has been deployed and is not in a blocked or suspended state, and if the previous fiscal day has been closed. The receipt creation process includes saving the data structure of the Cash Register Opening Receipt into the database and keeping track of the fiscal day's identifiers within the FAM.

API endpoint group details: FCU interface/Document – Receipt management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateFiscalDayOpen (DocCreateRequest leszármazottja)

Endpoint Response Objects: DocCreateDocumentResponse

Request data structure

```
{  
    "@type": "createFiscalDayOpen",  
    "systemId" : "{{systemId}}",  
    "cashDrawer" : [ {  
        "name": "Készpénz",  
        "moneyCat" : "CASH",  
        "moneySubCat" : null,  
        "moneyAmount" : "100000.00",  
        "currency" : "{{currency}}"  
    } ],  
}
```



```
"docCustomInfo": [{}  
    "@type": "text",  
    "key": "R00001_NAPNYIT_KIEG",  
    "description": "Leírás",  
    "text": "DAY OPEN CUSTOM INFO",  
    "alignment": "CENTER",  
    "orderId": 1  
],  
"attachment": ...,  
"createDownloadInfo": false,  
"submitKeyRequest": {  
    "publicKey": "...",  
    "date": 1715088749000,  
}  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM REST interface about the type of data structure submitted in the HTTP request (=createFiscalDayOpen).
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **cashDrawer** - The data structure for creating a fiscal day opening receipt allows specifying the opening cash balance in the request. Payment instruments must be listed separately by type. The data structure is described in the [PaymentDetails](#) section.
- **docCustomInfo** - Custom information related to the fiscal day opening receipt can be specified, with the data structure details found in the [CustomInfo \(item\)](#).
- **attachment** – Receipt attachment. The related data structure is described in the [Receipt Attachment](#) section.
- **createDownloadInfo** - A switch for generating a QR code containing receipt download information, which is either printed on the receipt copy or displayed on the screen.
The details are explained in the “[Formation of the output QR code for the E-cash register](#)” section.
- **submitKeyRequest** – A data structure used for transferring data scanned from a QR code generated by the customer application. The field details are explained in the [Data transfer from the customer application](#) section.

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "success",  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "documentData": [  
        {  
            "@type": "docCreate",  
            ...  
        }  
    ],
```



```
    "docDownloadInfo": "..."}  
}
```

Data structure field explanation:

- **fiscalDayNo** - The sequence number of the currently open fiscal day.
- **documentId** - The unique identifier of the created fiscal day opening receipt.
- **documentData** - The data required for displaying the receipt image (@type = docCreate).)
- **docDownloadInfo** - The content of the QR code printed on the receipt copy, containing the e-receipt download information.

The details are explained in the "[Formation of the output QR code for the E-cash register](#)" section.

The system first performs general request validations ([General request validations and response messages](#)) and then executes endpoint-specific checks:

Validation/Description	Result Code	J	Action Required
The fiscal day has already been opened earlier	FISCAL_DAY_OPEND_ALREADY	P	The continuation of fiscal operations assuming an open fiscal day, or the closure of the fiscal day. The error message can be prevented by performing a status query before attempting to open the day.

12.9.5 Receipt

The FAM is responsible for performing arithmetic tasks (summation of items) and carrying out payment-related calculations. The receipt includes a description of the purchased goods or services, prices, quantities, and the total amount. Additionally, business or other custom information can be attached.

The creation of the receipt document follows these steps:

- Creating the receipt, optionally with item entries
- Optionally adding any number of additional items to the document
- Closing the document by providing payment details

12.9.5.1 Receipt Creation

Creating a receipt is only possible within an open fiscal day. The process includes saving the receipt's data structure to the database and recording its identifiers within the fiscal day. The receipt creation data structure allows multiple receipt items to be specified in the request. In such cases, the FAM not only creates the document but also adds the specified items to the receipt.

API endpoint group details: FAM interface/Document – Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateReceipt (DocCreateRequest descendants)

Endpoint Response Objects: DocCreateDocumentResponse



Request data structure

```
{  
    "@type": "createReceipt",  
    "systemId": "{{systemId}}",  
    "receiptItems": [  
        {  
            "itemName": "Koktélparadicsom",  
            "itemArticleNo": "5998765676545",  
            "itemUnitPrice": "1499.00",  
            "itemQty": "1.0000",  
            "itemUnit": "KILOGRAM",  
            "itemCat": "SALE",  
            "itemDept": "A",  
            "itemCustomInfo": [  
                {  
                    "@type": "text",  
                    "key": "E00011_SZARM_HELY",  
                    "description": "Leírás",  
                    "text": "Szárm. hely: Magyarország",  
                    "alignment": "CENTER",  
                    "orderId": 1  
                }  
            ]  
        }  
    ]  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure submitted in the HTTP request (=createReceipt)
- **systemId*** The unique identifier of the FAM instance (AP number)
- **receiptItems** – The list of initially added items and the structural composition of each item. Providing an item is not mandatory for creation. The related data structure description can be found under the [receiptItems](#) section.

Response Data Structure (Upon Successful Execution)

```
{  
  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "docTotal": "1499.00",  
    "documentData" : [  
        {  
            "@type": "docCreate",  
            ...  
        },  
    ],  
}
```



```
{  
    "@type": "receiptItem",  
    "itemId": 1,  
    "itemSum": "1499.00",  
    "itemName": "Cherry tomato",  
    "itemArticleNo": "5998765676545",  
    "itemUnitPrice": "1499.00",  
    "itemQty": "1.0000",  
    "itemUnit": "KILOGRAM",  
    "itemCat": "SALE",  
    "itemDept": "A",  
    "itemCustomInfo": [  
        {  
            "@type": "text",  
            "key": "E00011_SZARM_HELY",  
            "description": "Leírás",  
            "orderId": 1,  
            "text": "comment",  
            "alignment": "CENTER"  
        }  
    ]  
},  
...  
]  
}
```

Data structure field explanation:

- **fiscalDayNo** - The sequence number of the currently open fiscal day
- **documentId** - The unique identifier of the receipt created
- **docTotal** – The current total amount of the document
- **documentData** – The data required to display the receipt image

12.9.5.2 Adding an item

An unlimited number of items can be registered for open sales receipts. When adding items, the FAM stores the item data structure in the database, performs the necessary arithmetic calculations, and increments the internal counters of the receipt. An item (receiptItem) can be added to sales receipts.

API endpoint group details: FAM interface/Document – Document management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/add-item

Endpoint Request Objects: AddReceiptItemRequest - When adding an item, the AddItemRequest descendants are used

Endpoint Response Objects: AddItemDocumentResponse



Request data structure when adding an item

```
{  
    "@type": "addReceiptItem",  
    "systemId": "{{systemId}}",  
    "documentId": {{documentId}},  
    "receiptItems": [  
        {  
            "itemName": "Favorit white bread",  
            "itemArticleNo": "5998576454321",  
            "itemUnitPrice": "399.00",  
            "itemQty": "1.0000",  
            "itemUnit": "PIECE",  
            "itemCat": "SALE",  
            "itemDept": "B",  
            "itemCustomInfo": [  
                {  
                    "@type": "text",  
                    "key": "E00001_TETEL_ISM",  
                    "description": "Leírás",  
                    "text": "comment",  
                    "alignment": "CENTER",  
                    "orderId": 1  
                }  
            ]  
        }  
    ]  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=addReceiptItem)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentId*** - The unique identifier of the created receipt
- **receiptItems*** - The data structure of items added to the receipt
The list of added items and the structural composition of each item. The related data structure description can be found under the [receiptItems](#) section.

The accepted values for the given item category (itemCat) in the case of a receipt are:

- SALE
- VOID_SALE
- DISCOUNT
- VOID_DISCOUNT
- NB_DISCOUNT
- VOID_NB_DISCOUNT
- SURCHARGE
- VOID_SURCHARGE
- EMPTIES
- VOID_EMPTIES

The field value set list and the explanation of each value can be found in the descriptive section above.



Response data structure in case of successful item addition

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "docTotal": "1898.00",  
    "documentData": [  
        {  
            "@type": "receiptItem",  
            "itemId": 2,  
            "itemSum": "399.00",  
            "itemName": "Favorit white bread",  
            "itemArticleNo": "5998576454321",  
            "itemUnitPrice": "399.00",  
            "itemQty": "1.0000",  
            "itemUnit": "PIECE",  
            "itemCat": "SALE",  
            "itemDept": "B",  
            "itemCustomInfo": [  
                {  
                    "@type": "text",  
                    "key": "E00001_TETEL_ISM",  
                    "description": "Leírás",  
                    "orderId": 1,  
                    "text": "comment",  
                    "alignment": "CENTER"  
                }  
            ]  
        }  
    ]  
}
```

Data structure field explanation:

- **fiscalDayNo** - The serial number of the currently open fiscal day
- **documentId** - The unique identifier of the receipt created
- **docTotal** - The current total amount of the receipt
- **documentData** - The data required to display the receipt image

12.9.5.3 Receipt closure

If items have been added to the opened receipt, the document must be closed. When closing the receipt, it is mandatory to provide payment information, which the FAM verifies and saves to the database. After the receipt is closed, adding further items to the receipt is not allowed.

The **interruption** of receipt creation by the operator also takes place at this endpoint.

API endpoint group details: FAM interface/Document – Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-close

Endpoint Request Objects:

- DocCloseReceipt - Receipt closure in case of a successful sale,



- DocCloseInterruption - Receipt closure in case of an interrupted sale (descendant of DocCloseRequest)
- Endpoint response object: *DocCloseResponse*

Request data structure for a successful sale

```
{  
    "@type": "closeReceipt",  
    "systemId": "{{systemId}}",  
    "documentId": {{documentId}},  
    "docCustomInfo": [  
        {  
            "@type": "text",  
            "key": "K01005_KOSZ",  
            "description": "Leírás",  
            "text": "Köszönjük a vásárlást!",  
            "alignment": "CENTER",  
            "orderId": 1  
        }  
    ],  
    "paymentDetails": [  
        {  
            "name": "Készpénz",  
            "moneyCat": "CASH",  
            "moneySubCat": null,  
            "moneyAmount": "5000",  
            "currency": "HUF"  
        }  
    ],  
    "serviceFee": "1000",  
    "attachment": ...  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - The FAM Document interface informs about the type of data structure sent in the HTTP request (=closeReceipt)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentId*** - The unique identifier of the created receipt
- **docCustomInfo** - The description of unique informational data that can be provided in custom information related to the receipt can be found in the structure of the [CustomInfo \(item\)](#) section.
- **paymentDetails*** - The paymentDetails data structure is described in the [PaymentDetails – Receipt Monetary Data](#) section.
- **serviceFee** – Service fee
- **attachment** - Receipt attachment

Request data structure for an interrupted sale

```
{  
    "@type": "closeInterruption",  
    "systemId": "{{systemId}}",  
    "documentDescriptor": {  
        "type": "RECEIPT",  
    },  
    "serviceFee": "1000",  
    "attachment": ...  
}
```



```
"docId": {{documentId}},
"fiscalDayNo": {{fiscalDayNo}},
"interrupted": true
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type***- Informs the FAM Document interface about the type of data structure sent in the HTTP response (=closeInterruption)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentDescriptor*** - The data structure identifying the receipts

The related data structure description can be found under the [documentDescriptor](#) section.

- **type*** = RECEIPT

The data required to display the receipt image

```
{
  "documentData": [
    {
      "@type": "docClose",
      ...
    }
  ],
  "resultCode": "SUCCESS",
  "resultDesc": null,
  "remainingSum": "0.00",
  "docDownloadInfo": "{...}"
}
```

Data structure field explanation:

- **documentData** – The data required to display the receipt image
- **remainingSum** - The remaining amount to be paid; in the case of exact payment, its value is always "0.00"
- **docDownloadInfo** – The QR code content to be printed on the receipt copy for the customer, containing information for downloading the e-receipt. The content is detailed in the [Formation of the output QR code for the E-cash register](#) section.

Response data structure in case of a successful document interruption

```
{
  "documentData": [
    {
      "@type": "receiptItem",
      ...
    },
    ...
    {
      "@type": "docClose",
      ...
    }
  ]
}
```



```
        }
    ],
    "resultCode": "SUCCESS",
    "resultDesc": null,
    "remainingSum": null
}
```

Data structure field explanation

- **documentData** – The data required to display the receipt image. Its content includes all previously added items with opposite signs, as well as the data structure of the closure.

The values of the other fields in the response are *null*.

The system performs general request validations ([General request validations and response messages](#)). There is no specific validation for the endpoint

12.9.6 Cancellation document

A cancellation document can only be issued for a receipt or a simplified invoice. The document must include the buyer's details or the identification number of the protocol justifying the issuance of the cancellation document, as well as the reason for cancellation. Business or other unique informational details can also be attached to the cancellation document.

The creation of a cancellation document follows these steps:

- Creation of the cancellation document
- Addition of any number of items to the document
- Closing the document with payment details

These steps can be passed to the FAM through multiple consecutive API calls.

12.9.6.1 Creating a cancellation document

The document can only be created within an open fiscal day. The creation process includes saving the cancellation document's data structure into the database and recording its identifiers within the fiscal day. The structure of the cancellation document allows for multiple receipt items to be specified in the request. In such cases, the FAM will not only create the document but also add the specified items to it.

API endpoint group details: FAM interface/Document – Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateVoidReceipt (descendant of DocCreateRequest)

Endpoint Response Objects: DocCreateDocumentResponse

Request data structure

```
{
    "@type": "createVoidReceipt",
    "systemId": "{{systemId}}",
    "sourceDocNo": "NY-A00000001/2000002/0001/00001",
    "voidReason": "S1",
    "sourceDocType": "RECEIPT",
    "fulfillmentDate": null,
```



```
"paymentDue": null,  
"paymentType": null,  
"billTo": null,  
"receiptItems": [  
  {  
    "itemRef": 3,  
    "itemName": "Cherry tomato",  
    "itemArticleNo": "5998765676545",  
    "itemUnitPrice": "1499.00",  
    "itemQty": "-1.0000",  
    "itemUnit": "KILOGRAM",  
    "itemCat": "VOID_SALE",  
    "itemDept": "A",  
    "itemCustomInfo": [  
      {  
        "@type": "text",  
        "key": "T20001_SZIN",  
        "description": "Leírás",  
        "text": "comment",  
        "alignment": "CENTER",  
        "orderId": 1  
      }  
    ]  
  }  
]
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** -) Informs the FAM Document interface about the type of data structure sent in the HTTP request (=createVoidReceipt)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **sourceDocNo*** The unique identifier of the original receipt or simplified invoice
- **voidReason*** - Reason for cancellation
- **sourceDocType*** - Original receipt type, the values can be:
 - **RECEIPT** - Receipt
 - **SIMPLE_INVOICE** – Simplified invoice
 - **INVOICE** – Invoice
- **fulfillmentDate** – Date of fulfillment, mandatory to fill in the case of an invoice
- **paymentDue** – Payment deadline, mandatory to fill in the case of an invoice
- **paymentType** –Method of payment, mandatory to fill in the case of an invoice and receipts cancelling/modifying it
 - **CASH** – Cash payment
 - **WIRE_TRANSFER** – Wire transfer
- **billTo** - Buyer's details (Mandatory for invoice cancellation and modification!), the data structure is described in the [CustomerData](#) section.
- **receiptItems** - The list of added items and the structural composition of each item. The corresponding data structure is described in [receiptItems](#) section.



- **voidReason - Possible values for the reason of cancellation:**

- **S1** - Customer withdrawal
- **S2** - Operator error: incorrect entry
- **S3** - Operator error: incorrect payment method entry
- **S4** - Operator error: product out of stock
- **S5** - Technical: incorrect document type issuance
- **S6** - Technical: failed payment method usage
- **S7** - Technical: incorrect customer data/entry error
- **S8** - Technical: test purchase
- **S0** - Other

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "docTotal": "1499.00",  
    "docTotalTax": null,  
    "docTotalNet": null,  
    "documentData": [  
        {  
            "@type": "docCreate",  
            ...  
        },  
        {  
            "@type": "receiptItem",  
            "itemRef": 3,  
            "itemId": 1,  
            "itemSum": "-1499.00",  
            "itemName": "Cherry tomato",  
            "itemArticleNo": "5998765676545",  
            "itemUnitPrice": "1499.00",  
            "itemQty": "-1.0000",  
            "itemUnit": "KILOGRAM",  
            "itemCat": "VOID_SALE",  
            "itemDept": "A",  
            "itemCustomInfo": [  
                {  
                    "@type": "text",  
                    "key": "T20001_SZIN",  
                    "description": "Leírás",  
                    "orderId": 1,  
                    "text": "comment",  
                    "alignment": "CENTER"  
                }  
            ]  
        },  
    ],  
}
```



Data structure field explanation:

- **fiscalDayNo** - The serial number of the currently open fiscal day
- **documentId** - The unique identifier of the created void receipt
- **docTotal** - The actual total amount of the receipt
- **docTotalTax** - The current VAT amount of the receipt (the data structure includes it only in the case of invoice cancellation)
- **dosTotalNet** - The current net amount of the receipt (the data structure includes it only in the case of invoice cancellation)
- **documentData** - The data required to display the receipt image

12.9.6.2 Adding an item

The items listed on the original receipt or void receipt can be added to the document. During the item addition process, the FAM saves the item data structure into the database, performs the necessary arithmetic calculations, and increments the internal counters of the receipt. Only one type of item can be added to sales receipts, including void receipts:

- Item/product to be voided

API endpoint group details: FAM interface/Document – Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/add-item

Endpoint Request Objects: AddVoidReceiptItemRequest

Endpoint Response Objects: AddItemDocumentResponse

Request data structure in case of successful item addition

Apart from the @type field, there is no structural difference compared to adding an item to a receipt.

```
{  
    "@type": "addVoidReceiptItem",  
    "systemId": "{{systemId}}",  
    "documentId": {{documentId}},  
    "receiptItems": [{  
        "itemRef": 1,  
        "itemName": "Favorit white bread",  
        "itemArticleNo": "5998576454321",  
        "itemUnitPrice": "399.00",  
        "itemQty": "-1.0000",  
        "itemUnit": "PIECE",  
        "itemCat": "VOID_SALE",  
        "itemDept": "B",  
        "itemCustomInfo": [  
            {  
                "@type": "text",  
                "key": "E00001_TETEL_ISM",  
                "description": "Leírás",  
                "text": "comment",  
                "alignment": "CENTER",  
                "orderId": 1  
            }  
        ]  
    }]
```



]
}

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=addVoidReceiptItem)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentId*** - The identifier of the void receipt
- **receiptItems*** - The data structure of the item added to the void receipt
The list of added items and the structural composition of each item. The related data structure is described under the [receiptItems](#) section. The specific usage of individual fields for this type of receipt is as follows:
 - **itemRef*** - The sequence number of the item on the original receipt (itemId)

The accepted values for the item category (itemCat) in the case of a void receipt:

- SALE
- VOID_SALE
- DISCOUNT
- VOID_DISCOUNT
- NB_DISCOUNT
- VOID_NB_DISCOUNT
- SURCHARGE
- VOID_SURCHARGE
- EMPTIES
- VOID_EMPTIES

Response data structure in case of successful item addition

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "docTotalNet": null,  
    "docTotal": "1898.00",  
    "docTax": null,  
    "documentData": [  
        {  
            "@type": "receiptItem",  
            "itemId": 2,  
            "itemSum": "-399.00",  
            "itemName": "Favorit white bread",  
            "itemArticleNo": "5998576454321",  
            "itemUnitPrice": "399.00",  
            "itemQty": "-1.0000",  
            "itemUnit": "PIECE",  
            "itemCat": "VOID_SALE",  
            "itemDept": "B",  
            "itemCustomInfo": [  
                {  
                    "@type": "text",  
                    "key": "E00001_TETEL_ISM",  
                    "description": "Leírás",  
                    "orderId": 1,  
                }  
            ]  
        }  
    ]  
}
```



```
        "text": "comment",
        "alignment": "CENTER"
    }
]
}
}
```

Data structure field explanation:

- **fiscalDayNo** - The serial number of the currently open fiscal day
- **documentId** - The identifier of the opened void receipt
- **docTotalNet** - Current net total amount of the receipt, only in case of invoice cancellation
- **docTotal** - The current total amount of the void receipt
- **docTotalTax** - Current VAT value of the receipt, only in the case of invoice cancellation
- **documentData** - The data required to display the receipt image

12.9.6.3 Closing the receipt

If items have been added to the opened void receipt, the receipt must be closed. During the closing of the receipt, it is mandatory to provide payment information, which is validated by FAM and stored in the database. Once the void receipt is closed, adding further items to the void receipt is not allowed.

The cancellation of the receipt process by the operator is also performed at this endpoint.

API endpoint group details: FAM interface/Document – Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-close

Endpoint Request Objects: (descendants of DocCloseRequest)

- DocCloseVoidReceipt - Void receipt closure in case of successful transaction,
- DocCloseInterruption - Void receipt closure in case of interrupted transaction

Endpoint response object: DocCloseResponse

Request data structure for successful transaction

Apart from the @type field, there is no structural difference compared to the Receipt closure process.

```
{
    "@type": "closeVoidReceipt",
    "systemId": "{{systemId}}",
    "documentId": "{{documentId}}",
    "docCustomInfo": [
        {
            "@type": "text",
            "key": "K00001_KOSZI",
            "description": "Leírás",
            "text": "Köszönjük a vásárlást!",
            "alignment": "CENTER",
        }
    ]
}
```



```
        "orderId": 1
    }
],
"paymentDetails": [
    {
        "name": "Készpénz",
        "moneyCat": "CASH",
        "moneySubCat": null,
        "moneyAmount": "-5000",
        "currency": "HUF"
    }
],
"serviceFee": null,
"attachment": ...
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=closeVoidReceipt)
- The other fields are the same as those defined in the [Receipt closure](#) section.

Request data structure in case of interrupted sales

```
{
    "@type": "closeInterruption",
    "systemId": "{{systemId}}",
    "documentDescriptor": {
        "type": "VOID_RECEIPT",
        "docId": {{documentId}},
        "fiscalDayNo": {{fiscalDayNo}},
        "interrupted": true
    }
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=closeInterruption)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentDescriptor*** - The data structure identifying the receipts
 - The related data structure description can be found under the [documentDescriptor](#) section.
 - **type*** = VOID_RECEIPT

The data required to display the receipt image

```
{
    "resultCode": "SUCCESS",
    "resultDesc": null,
    "remainingSum": "0.00",
    "documentData": [
        {
            "@type": "docClose",
            ...
        }
    ]
}
```



```
        ],
        "docDownloadInfo": "{...}"
    }
```

Data structure field explanation:

- **remainingSum** - Outstanding amount to be paid, always "0.00" in case of exact payment.
- **documentData** - The data required to display the receipt image
- **docDownloadInfo** – The content of the QR code to be printed on the receipt copy for the customer, containing the e-receipt download information.

The details are explained in the [Formation of the output QR code for the E-cash register](#) section.

Data Structure in Case of Successful Receipt Interruption

```
{
    "documentData": [
        {
            "@type": "receiptItem",
            ...
        },
        ...
        {
            "@type": "docClose",
            ...
        }
    ],
    "resultCode": "SUCCESS",
    "resultDesc": null,
    "remainingSum": null
}
```

Data structure field explanation:

- **documentData** - The data required to display the receipt image. Contains all previously added items with opposite signs, as well as the data structure of the closing resultCode - Identification code of the result of the task
- All other fields in the response will have a *null* value.

Specific Result Codes for the Endpoint:

The system first performs general request validations ([General request validations and response messages](#)), followed by specific endpoint checks:

Validation/Description	Result Code	J	Action Required
The receipt number (<i>sourceDocNo</i>) referenced in the request for opening the receipt is incorrect	INVALID_DOCUMENT_TYPE	P	Continue tax-related operations assuming an open fiscal day or close the fiscal day. The error message can be prevented by performing a status query before attempting to open the fiscal day

12.9.7 Money movement receipt

The money movement receipt is a document that records the financial transactions related to money movements, indicating the amounts involved in the transaction. The receipt must specify the reason for the money movement. There are no restrictions on the types of payment instruments that can be specified. It has three types:



- Payment
- Deposit
- Simultaneous payment and deposit:
 - Payment instrument exchange
 - Cash withdrawal (cash back)

Money Movement Receipt is created with a single API call

12.9.7.1 Creating money movement receipt

The Money Movement Receipt can be created at any time within an open fiscal day. The creation process includes saving the money movement receipt data structure to the database, registering the money movement recorded on the receipt, and incrementing the document counters within the fiscal day.

API endpoint group details: FAM interface/Document – Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateCashFlowReport (descendant of DocCreateRequest)

Endpoint Response Objects: DocCreateDocumentResponse

Request data structure

{

```
"@type": "createCashFlowReport",
"systemId": "{{systemId}}",
"pmtReason": "CHANGE",
"otherTitle": null,
"cashDrawer": [
    {
        "moneyCat": "CASH",
        "name": "Készpénz",
        "moneySubCat": null,
        "moneyAmount": "4000.00",
        "currency": "{{currency}}"
    }
],
"docCustomInfo": [
    {
        "@type": "text",
        "key": "F00101_CF",
        "description": "Leírás",
        "text": "CASH FLOW REPORT INFO",
        "alignment": "CENTER",
        "orderId": 1
    }
],
"attachment": ...,
"createDownloadInfo": false,
"submitKeyRequest": {
```



```
        "publicKey": "...",
        "date": 1715088749000,
    }
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure submitted in the HTTP request (=createCashFlowReport)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **pmtReason*** - The legal title of the receipt, specifically for cash deposit, withdrawal, or payment instrument exchange
- **otherTitle** - Must be filled in for other deposit or withdrawal cases
- **cashDrawer*** - Data on changes in cash drawer contents by payment instrument, the data structure is described in the PaymentDetails – Receipt Amount Data section.
- **docCustomInfo** - Custom information related to the money movement receipt, the description of the custom data structure can be found in the [CustomInfo](#) section.
- **attachment** –Receipt attachment
- **createDownloadInfo** - A switch for generating a QR code containing e-receipt download information, which must be printed on the receipt copy for the customer. The content is detailed in the "[Formation of the output QR code for the E-cash register](#)" section.
- **submitKeyRequest** - A data structure used to transfer data read from the customer application. The fields are explained under the "[Data transfer from the customer application](#)" subsection.

The legal title of the receipt (pmtReason) field value set:

- **Deposit:**
 - **CHANGE** - Change deposit
 - **CASHIER_DEPOSIT** - Cashier cash deposit
 - **COLLECTION** – Fee collection
 - **LOTTERY_TICKET_SALE** - Lottery ticket sale
 - **DEPOSIT** – Advance payment
 - **DEFICIT** – Cash register shortage
 - **TIP** - Tip
 - **OTHER_CASH_IN** -Other deposit (in this case, an additional optional field is expected: *otherTitle*)
- **Withdrawal:**
 - **SKIMMING** – Skimming
 - **CASHIER_WITHDRAWAL** - Cashier withdrawal
 - **VOUCHER_OUT** - Voucher withdrawal
 - **GIFT_CARD_OUT** - Gift card withdrawal
 - **SALARY_PAYMENT** – Salary payment
 - **ADVANCE_PAYMENT** - Salary advance
 - **POSTAL_FEE** – Postal fee



- **OTHER_FEES** - Other utility expenses
- **PROCUREMENT** – Goods purchase
- **TURNOVER_WITHDRAWAL** - Closing balance withdrawal
- **OTHER_CASH_OUT** -Other withdrawal (in this case, an additional optional field is expected: *otherTitle*)

Simultaneous deposit and withdrawal:

- **CURRENCY_EXCHANGE** – Payment instrument exchange
- **CASHBACK** – Cash withdrawal

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "success",  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "documentData": [  
        {  
            "@type": "docCreate",  
            ...  
        },  
        {  
            "@type": "reportItem",  
            ...  
        },  
        ...  
        {  
            "@type": "docClose",  
            ...  
        }  
    ],  
    "docDownloadInfo": null  
}
```

Data structure field explanation:

- **fiscalDayNo**- The serial number of the currently open fiscal day
- **documentId**- The unique identifier of the created cash flow receipt
- **documentData** - The data required to display the receipt image

The system performs general request validations ([General request validations and response messages](#)). In the case of disbursement, cash withdrawal, and currency exchange, the FAM verifies the cash drawer contents.

12.9.8 Cash register report

This document type provides information about the status of the fiscal day. The Cash Register Report receipt can be generated at any time during an open fiscal day. It is mandatory to generate it immediately before the Daily Turnover Report receipt.

The Cash Register Report receipt is generated through a single API call.



12.9.8.1 Cash register report receipt creation

The creation of the Cash Register Report receipt can be performed at any time within an open fiscal day. The process of creating the receipt includes saving the data structure of the Cash Register Report receipt into the database and incrementing the document counters within the fiscal day.

API endpoint group details: FAM interface/Document – Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateCashRegisterReport (descendant of DocCreateRequest)

Endpoint Response Objects: DocCreateDocumentResponse

Request data structure

```
{  
    "@type": "createCashRegisterReport",  
    "systemId": "{{systemId}}",  
    "docCustomInfo": [  
        {  
            "@type": "text",  
            "key": "R20001_CRR",  
            "description": "Leírás",  
            "text": "CASH REGISTER REPORT CUSTOM INFO",  
            "alignment": "CENTER",  
            "orderId": 1  
        }  
    ],  
    "attachment": ... ,  
    "createDownloadInfo": false,  
    "submitKeyRequest": {  
        "publicKey": "...",  
        "date": 1715088749000,  
    }  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure submitted in the HTTP request (=createCashRegisterReport)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **docCustomInfo** - Custom information that can be linked to the cash register report receipt. The structure of the custom information data is described in the [CustomInfo](#) section.
- **attachment** - Receipt attachment
- **createDownloadInfo** - A toggle for generating a QR code containing the e-receipt download information, which will be printed on the receipt copy for the customer.

The content of this QR code is explained in the chapter on “[Formation of the output QR code for the E-cash register](#)”



- **submitKeyRequest** - Data structure used for transmitting data read from the customer application. The details of its fields are explained in the "[Data transfer from the customer application](#)" subsection.

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "success",  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "documentData": [  
        {  
            "@type": "docCreate",  
            ...  
        },  
        {  
            "@type": "reportItem",  
            ...  
        },  
        ...,  
        {  
            "@type": "docClose",  
            ...  
        }  
    ],  
    "docDownloadInfo": null  
}
```

Data structure field explanation:

- **fiscalDayNo** - The serial number of the currently open fiscal day
- **documentId** - The unique identifier of the created cash register report receipt
- **documentData** - The data required to display the receipt image
- **docDownloadInfo** - The content of the QR code containing the e-receipt download information, which is to be printed on the receipt copy for the customer.
The content is explained in the chapter on "[Formation of the output QR code for the E-cash register](#)".

The system performs general request validations ([General request validations and response messages](#)). There are no specific validations for this endpoint.

12.9.9 Daily sales report

The Daily Sales Report is required to close the fiscal day. Before creating it, a Cash Register Report receipt must be generated.

The Daily Sales Report is generated with a single API call.

12.9.9.1 Creating the daily sales report

The Daily Sales Report receipt can be created at any time within an open fiscal day. Before its creation, generating a Cash Register Report receipt is mandatory. The process of creating the receipt includes saving the Daily Sales Report data structure into the database.



API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateFiscalDayReport (descendant of DocCreateRequest)

Endpoint Response Objects: DocCreateDocumentResponse

Request data structure

```
{  
    "@type": "createFiscalDayReport",  
    "systemId": "{{systemId}}",  
    "docCustomInfo": [  
        {  
            "@type": "text",  
            "key": "R05044_FISCALDAYCLOSE",  
            "description": "Leírás",  
            "text": "DAY CLOSE CUSTOM INFO",  
            "alignment": "CENTER",  
            "orderId": 1  
        }  
    ],  
    "attachment": ...,  
    "createDownloadInfo": false,  
    "submitKeyRequest": {  
        "publicKey": "...",  
        "date": 1715088749000,  
    }  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=createFiscalDayReport)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **docCustomInfo** - Custom information associated with the daily transaction report receipt. The description of custom information data fields can be found in the [CustomInfo](#) section.
- **attachment** – Receipt attachment
- **createDownloadInfo** - A switch for generating a QR code that contains e-receipt download information for the customer, which should be printed on the receipt copy. The details are explained in the "[Formation of the output QR code for the E-cash register](#)" section.
- **submitKeyRequest** - The explanation of the fields is provided in the "[Data transfer from the customer application](#)" subsection.

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "success",  
    "fiscalDayNo": 1,  
    "documentId": 1,
```



```
"documentData": [
    {
        "@type": "docCreate",
        ...
    },
    {
        "@type": "reportItem",
        ...
    },
    ...,
    {
        "@type": "docClose",
        ...
    }
],
"docDownloadInfo": null
}
```

Data structure field explanation:

- **fiscalDayNo** - The serial number of the currently open fiscal day
- **documentId** - the unique identifier of the created daily transaction report
- **documentData** - The data required to display the receipt image
- **docDownloadInfo** - The content of the QR code to be printed on the receipt copy for the customer, containing the e-receipt download information. The details are explained in the "[Formation of the output QR code for the E-cash register](#)" section.

Specific Result Codes for the Endpoint:

The system first performs general request validations ([General request validations and response messages](#)), and then executes endpoint-specific checks:

Validation/Description	Result Code	J	Action Required
The Cash Register Report receipt was not created before the daily closing.	CASH_REGISTER_REPORT_IS_NOT_CREATE_D	P	A Cash Register Report receipt must be created.

12.9.10 Modification receipt

A modification receipt is issued when, in accordance with the conditions set out in the purchase contract between the buyer and the supplier, the buyer has the right to return or bring back the goods. The modification receipt is a crucial document as it facilitates the process of returning goods and issuing refunds. It contains details of the original purchase, including the unique identifier of the original receipt or simplified invoice, as well as the details of the product and the buyer. Additionally, it includes the reason for the modification, explaining why the goods were returned, and business-related or other specific custom information can also be attached.

The process of issuing a modification receipt includes:

- Creating the modification receipt
- Adding any number of items to the receipt
- Closing the receipt by providing payment information

These steps can be passed to the FAM through multiple consecutive API calls.



12.9.10.1 Creating a modification receipt

A modification receipt can only be created within an open fiscal day. The process includes saving the modification receipt's data structure to the database and recording its identifiers within the fiscal day. During creation, the buyer's data, the identifying fields of the original receipt, and the reason for the modification are stored. The modification receipt's data structure allows multiple items to be modified within the same HTTP request.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateReturnReceipt (descendant of DocCreateRequest)

Endpoint Response Objects: DocCreateDocumentResponse

Request data structure

```
{
    "@type": "createReturnReceipt",
    "systemId": "{{systemId}}",
    "sourceDocNo": "NY-A00000001/2000002/0001/0001",
    "returnReason": "V1",
    "sourceDocType": "RECEIPT",
    "fulfillmentDate": null,
    "paymentDue": null,
    "paymentType": null,
    "receiptItems": [
        {
            "itemRef": 1,
            "itemName": "Cherry tomato",
            "itemArticleNo": "5998765676545",
            "itemUnitPrice": "1499.00",
            "itemQty": "-1.0000",
            "itemUnit": "KILOGRAM",
            "itemCat": "RETURN",
            "itemDept": "A",
            "itemCustomInfo": [
                {
                    "@type": "text",
                    "key": "T80001_FURTOS",
                    "description": "Leírás",
                    "text": "comment",
                    "alignment": "CENTER",
                    "orderId": 1
                }
            ]
        }
    ]
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure submitted in the HTTP request (=createReturnReceipt)
- **systemId*** - The unique identifier of the FAM instance (AP number)



- **sourceDocNo*** - The identifier of the original receipt (a free-text field to allow specifying unique receipt identifiers)
- **returnReason*** - The reason for the modification:
 - **V1** - Defective goods
 - **V2** - Customer withdrawal from the purchase
 - **V3** - Other (including returnable packaging redemption)
- **sourceDocType** – original document type, values can be
 - **RECEIPT** - receipt
 - **SIMPLE_INVOICE** - simplified invoice
 - **INVOICE** - invoice.
- **fulfillmentDate** – date of fulfillment, mandatory in case of invoice
- **paymentDue** - Payment deadline, mandatory to fill in the case of an invoice
- **paymentType** - Method of payment, mandatory to fill in the case of an invoice and receipts cancelling/modifying it
 - **CASH** – cash
 - **WIRE_TRANSFER** – wire transfer
- **receiptItems** - The structural details of the item, elaborated in the [receiptItems](#) and [Adding Items](#) subsection.

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "docTotal": "1499.00",  
    "docTotalTax": null,  
    "docTotalNet": null,  
    "documentData": [  
        {  
            "@type": "docCreate",  
            ...  
        },  
        {  
            "@type": "receiptItem",  
            "itemId": 1,  
            "itemSum": "-1499.00",  
            "itemName": "Cherry tomato",  
            "itemArticleNo": "5998765676545",  
            "itemUnitPrice": "1499.00",  
            "itemQty": "-1.0000",  
            "itemUnit": "kg",  
            "itemCat": "RETURN",  
            "itemDept": "A",  
            "itemCustomInfo": [  
                {  
                    "@type": "text",  
                    "key": "T80001_FURTOS",  
                    "description": "Leírás",  
                    "orderId": 1,  
                    "text": "comment",  
                    "alignment": "CENTER"  
                }  
            ]  
        }  
    ]
```



```
        }
    ]
}
```

Data structure field explanation:

- **fiscalDayNo** - The serial number of the currently open fiscal day
- **documentId** - Unique Identifier of the Created Modification Receipt
- **docTotal** – The current total amount of the receipt
- **docTotalTax** – The current VAT amount of the receipt (the data structure includes it only in the case of invoice cancellation)
- **docTotalNet** - The current net amount of the receipt (the data structure includes it only in the case of invoice cancellation)
- **documentData** - The data required to display the receipt image

12.9.10.2 Adding an item

When adding items, the FAM saves the item's data structure in the database, performs the necessary arithmetic calculations, and increments the internal counters of the receipt. The items in the modification receipt can be those that appeared on the original receipt, but the FAM does not verify the original receipt item by item.

For sales receipts, including modification receipts, only one type of item can be added:

- Item/Product to be modified

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/add-item

Endpoint Request Objects: AddReturnReceiptItemRequest - When adding an item, the following request data structure is used, which inherits from AddItemRequest)

Endpoint Response Objects: (AddItemDocumentResponse

Request data structure when adding an item

```
{
    "@type": "addReturnReceiptItem",
    "systemId": "{{systemId}}",
    "documentId": {{documentId}},
    "receiptItems": [
        {
            "itemRef": 5,
            "itemName": "Favorit white bread",
            "itemArticleNo": "5998576454321",
            "itemUnitPrice": "399.00",
            "itemQty": "-1.0000",
            "itemUnit": "PIECE",
            "itemCat": "RETURN",
            "itemDept": "B",
            "itemCustomInfo": [
                {
                    "@type": "text",
                    "key": "B12121_GLUTEN",
                    "description": "Leírás",
                    "text": "comment",
                    "alignment": "CENTER",
                    "fontSize": 12
                }
            ]
        }
    ]
}
```



```
        "orderId": 1
    }
]
}
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure submitted in the HTTP request (=addReturnReceiptItem)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentId*** - The unique identifier of the modification receipt
- **receiptItems*** - The data structure of the item added to the modification receipt. The description of the related data structure can be found under the [receiptItems](#) section. For this type of receipt, the specific usage of each field is as follows:
 - **itemRef*** - The serial number of the item on the original receipt (itemId), mandatory in the case of a voiding receipt

Handling of Item Categories (itemCat) in a Modification Receipt

The system determines how item categories should be treated based on the sourceDocType value specified in the HTTP request.

The FAM system maps the item category from the original receipt type and assigns the correct value for handling the modification receipt.

Possible itemCat values for modification receipts:

- **RETURN** - „v”: Return item
- **VOID_RETURN** - „vs”: Void return item
- **EMPTIES** - „g”: Deposit return
- **VOID_EMPTIES** - „gs”: Void deposit return
- **SURCHARGE** - „f”: Additional charge
- **VOID_SURCHARGE** - „fs”: Void additional charge
- **DISCOUNT** - „e”: Discount applied
- **VOID_DISCOUNT** - „es”: Void discount
- **NB_DISCOUNT** - „k”: non-commercial discount
- **VOID_NB_DISCOUNT** - „ks”: non-commercial discount cancellation

Response data structure in case of successful item addition

```
{
  "resultCode": "SUCCESS",
  "resultDesc": null,
  "fiscalDayNo": 1,
  "documentId": 1,
  "docTotalNet": null,
  "docTotal": "1898.00",
  "docTax": null,
  "documentData": [
    {
      "@type": "receiptItem",
      "itemId": 2,
      "itemSum": "-399.00",
    }
  ]
}
```



```
"itemName": "Favorit white bread",
"itemArticleNo": "5998576454321",
"itemUnitPrice": "399.00",
"itemQty": "-1.0000",
"itemUnit": "PIECE",
"itemCat": "RETURN",
"itemDept": "B",
"itemCustomInfo": [
    {
        "@type": "text",
        "key": "B12121_GLUTEN",
        "description": "Leírás",
        "orderId": 1,
        "text": "comment",
        "alignment": "CENTER"
    }
]
}
]
```

Data structure field explanation:

- **@type** - The type of the response data structure
- **fiscalDayNo** - The serial number of the currently open fiscal day
- **documentId** - The unique identifier of the opened modification receipt
- **docTotalNet** - The current net amount of the receipt (the data structure includes it only in the case of invoice cancellation)
- **docTotal** – The current total amount of the receipt
- **docTotalTax** – The current VAT amount of the receipt (the data structure includes it only in the case of invoice cancellation)
- **documentData** - The data required to display the receipt image

12.9.10.3 Closing the receipt

If items have been added to the opened modification receipt, the receipt must be closed. When closing the receipt, it is mandatory to provide payment information, the correctness of which is verified and stored in the database by the FAM. Once the modification receipt is closed, adding further items to the receipt is not allowed.

The cancellation of the receipt creation by the operator also occurs at this endpoint.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-close

Endpoint Request Objects: DocCloseReturnReceipt - Closing the modification receipt in case of a successful transaction.

DocCloseInterruption - Closing the modification receipt in case of an interrupted transaction. (Descendants of DocCloseRequest).

Endpoint Response Objects: DocCloseResponse



Request data structure in case of successful sales

```
{  
    "@type": "closeReturnReceipt",  
    "systemId": "{{systemId}}",  
    "documentId": {{documentId}},  
    "docCustomInfo": [  
        {  
            "@type": "text",  
            "key": "K10001_KOSZIKE",  
            "description": "Leírás",  
            "text": "Köszönjük a vásárlást!",  
            "alignment": "CENTER",  
            "orderId": 1  
        }  
    ],  
    "paymentDetails": [  
        {  
            "name": "Készpénz",  
            "moneyCat": "CASH",  
            "moneySubCat": null,  
            "moneyAmount": "-1900",  
            "currency": "HUF"  
        }  
    ],  
    "serviceFee": null,  
    "attachment": ...  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - The FAM Document interface informs about the type of data structure sent in the HTTP request (=closeReturnReceipt)
- The rest of the fields are identical to those defined in the Receipt closure description.

Request data structure in case of an interrupted transaction

```
{  
    "@type": "closeInterruption",  
    "systemId": "{{systemId}}",  
    "documentDescriptor": {  
        "type": "RETURN_RECEIPT",  
        "docId": {{documentId}},  
        "fiscalDayNo": {{fiscalDayNo}},  
        "interrupted": true  
    }  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - The FAM Document interface informs about the type of data structure sent in the HTTP request (=closeInterruption)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentDescriptor*** - The data structure identifying the receipts
 - **type*** = RETURN_RECEIPT
 - **docId*** - The identifier of the receipt



- **fiscalDayNo*** - The serial number of the fiscal day
- **interrupted*** - A flag indicating the interruption of the receipt

The data required to display the receipt image

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
  
    "documentData": [  
        {  
            "@type": "docClose",  
            ...  
        }  
    ],  
    "docDownloadInfo": "{...}"  
}
```

Data structure field explanation:

- **remainingSum** - Remaining amount to be paid, always „0.00” in the case of exact payment
- **documentData** - The data required to display the receipt image
- **docDownloadInfo** – The content of the QR code to be printed on the receipt copy for the customer, containing the e-receipt download information.

Response data structure in case of successful receipt interruption

```
{  
    "documentData": [  
        {  
            "@type": "receiptItem",  
            ...  
        },  
        ...  
        {  
            "@type": "docClose",  
            ...  
        }  
    ],  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "remainingSum": null  
}
```

Data structure field explanation

- **documentData** - The data required to display the receipt image. It contains all previously added items with opposite signs, as well as the closing data structure. All other fields in the response have a *null* value.

The system first performs general request validations, ([General request validations and response messages](#)) then carries out endpoint-specific validations:



Validation/Description	Result Code	J	Action Required
The document number (<i>sourceDocNo</i>) referenced in the document opening request is incorrect	INVALID_DOCUMENT_TYPE	P	Continue fiscal operations, assuming an open fiscal day, or close the fiscal day. This error message can be prevented by performing a status query before attempting to open the fiscal day.

12.9.11 Receipt summary report

The receipt summary report provides information on the fiscal receipts issued during the most recently opened fiscal day and any other fiscal days opened within the given calendar day. The report data includes counters for sales receipts created within the fiscal days, as well as individual receipt details categorized by receipt type. If there are no previous fiscal day openings associated with the given FAM instance outside of the currently open fiscal day, the report will only contain data for the currently open day.

12.9.11.1 Creating a receipt summary report

The creation of the receipt summary report can be performed at any time within an open fiscal day. The process includes saving the receipt summary data structure in the database and incrementing document counters within the fiscal day.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateReceiptListReport (descendants DocCreateRequest)

Endpoint Response Objects: DocCreateDocumentResponse

Request data structure

```
{  
    "@type": "createReceiptListReport",  
    "systemId": "{{systemId}}",  
    "attachment": ... ,  
    "createDownloadInfo": false,  
    "submitKeyRequest": {  
        "publicKey": "...",  
        "date": 1715088749000,  
    }  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure submitted in the HTTP request (=createReceiptListReport)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **attachment** – Receipt attachment
- **createDownloadInfo** - A toggle for generating a QR code containing the e-receipt download information, which will be printed on the receipt copy for the customer. The content is explained in the section [Formation of the output QR code for the E-cash register](#) section.



- **submitKeyRequest** - Data structure used for transferring data read from the customer application. The detailed field explanations can be found under the [Data transfer from the customer application](#) subsection.

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "success",  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "documentData": [  
        {  
            "@type": "docCreate",  
            ...  
        }  
    ],  
    "docDownloadInfo": "{...}"  
}
```

Data structure field explanation:

- **fiscalDayNo** - The serial number of the currently open fiscal day
- **documentId** - The unique identifier of the created receipt summary
- **documentData** - The data required to display the receipt image
- **docDownloadInfo** - The content of the QR code to be printed on the receipt copy for the customer, containing the e-receipt download information. The content is explained in the section [Formation of the output QR code for the E-cash register](#) section.

12.9.12 Simplified invoice

The simplified invoice contains customer details, a description of the purchased goods or services, prices, quantities, and the total amount. Additionally, business or other custom information can be attached.

The simplified invoice can be either electronic or paper based. In the latter case, printing two copies of the invoice is mandatory. It is important that a paper-based invoice should only be initiated if the successful printing is ensured. In such cases, it is recommended to check the printer connection in advance or offer the user a print test.

The simplified invoice is created in the following steps:

- Creating the simplified invoice
- Adding any number of items to the invoice
- Closing the invoice by providing payment details

These steps can be executed through multiple consecutive API calls to the FAM.



12.9.12.1 Creating a simplified invoice

The creation of a simplified invoice is only possible within an open fiscal day. The process includes saving the simplified invoice's data structure in the database, storing customer data, and maintaining the identifiers of the simplified invoice within the fiscal day.

The creation of a simplified invoice is essentially the same as handling a receipt, with the only difference being that customer details must be provided in the HTTP request when creating the simplified invoice.

A receipt that has already been opened can be modified into a simplified invoice using the **changeDocType** call, as described in the [Modification of the open receipt type](#).

The customer can choose whether they want a paper-based simplified invoice. In this case, the billTo field's invoiceType must be set to PAPER when submitted.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateSimpleInvoice (descendant of DocCreateRequest)

Endpoint Response Objects: DocCreateDocumentResponse

Request data structure

```
{  
    "@type": "createSimpleInvoice",  
    "systemId": "{{systemId}}",  
    "fulfillmentDate": 1720519280905,  
    "billTo": {  
        "name": "Teszt Elek",  
        "address": {  
            "addressType": "SIMPLE",  
            "countryCode": "HU",  
            "postCode": "1000",  
            "city": "Budapest",  
            "additionalAddressDetail": "Vas utca 33"  
        },  
        "taxNumber": {  
            "taxpayerId": "30000003",  
            "vatCode": "3",  
            "countyCode": "33"  
        },  
        "groupMemberTaxNumber": null,  
        "communityTaxNumber": null,  
        "thirdCountryTaxNumber": null,  
        "customerVatStatus": "DOMESTIC",  
        "invoiceType": "ELECTRONIC",  
        "bankAccountNo": "123123123123123123"  
    },  
    "receiptItems": [{  
        "itemName": "Cherry tomato",  
        "itemArticleNo": "5998765676545",  
        "itemUnitPrice": "1499.00",  
        "itemQty": "1.0000",  
    }],  
}
```



```
"itemUnit": "KILOGRAM",
"itemCat": "SALE",
"itemDept": "A",
"itemCustomInfo": [
    {
        "@type": "text",
        "key": "T80001_FURTOS",
        "description": "Leírás",
        "text": "comment",
        "alignment": "CENTER",
        "orderId": 1
    }
]
}]
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=createSimpleInvoice)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **fulfillmentDate** – Date of performance. It is mandatory to provide if its value differs from the date of the open fiscal day.
- **billTo*** - Customer details, the data structure is described in the [billTo](#) section.

Response Data Structure (Upon Successful Execution)

```
{
    "resultCode": "SUCCESS",
    "resultDesc": null,
    "fiscalDayNo": 1,
    "documentId": 1,
    "docTotal": "1499.00",
    "documentData": [
        {
            "@type": "docCreate",
        },
        {
            "@type": "receiptItem"
        }
    ]
}
```

Data structure field explanation:

- **fiscalDayNo** - The serial number of the currently open fiscal day
- **documentId** - The unique identifier of the created simplified invoice
- **docTotal** - The current total amount of the document
- **documentData** - The data required to display the receipt image

12.9.12.2 Adding an item

Any number of items can be registered for open sales documents. When adding items, the FAM stores the item data structure in the database, performs the necessary arithmetic calculations, and increments the internal counters of the simplified invoice. Items (**receiptItem**) can be added to sales documents.



API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/add-item

Endpoint Request Objects: AddSimpleInvoiceItemRequest – in case of adding an item
(descendants of AddItemRequest)

Endpoint Response Objects: AddItemDocumentResponse

Request Data Structure for adding an item

```
{  
    "@type": "addSimpleInvoiceItem",  
    "systemId": "{{systemId}}",  
    "documentId": {{documentId}},  
    "receiptItems": [  
        {  
            "itemName": "Favorit white bread",  
            "itemArticleNo": "5998576454321",  
            "itemUnitPrice": "399.00",  
            "itemQty": "1.0000",  
            "itemUnit": "PIECE",  
            "itemCat": "SALE",  
            "itemDept": "B",  
            "itemCustomInfo": [  
                {  
                    "@type": "text"  
                    "key": "F67071_GF",  
                    "description": "Leírás",  
                    "text": "comment",  
                    "alignment": "CENTER",  
                    "orderId": 1  
                }  
            ]  
        }  
    ]  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=addSimpleInvoiceItem)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentId*** - The identifier of the simplified invoice
- **receiptItems*** - The data structure of the item added to the simplified invoice. The related data structure is described in the [receiptItems](#) section.

Possible values for the item type (*itemCat*) in the case of a simplified invoice:

- **SALE** - „n”: sales
- **VOID_SALE** - „ns”: voided Sale
- **DISCOUNT** - „e”: discount
- **VOID_DISCOUNT** - „es”: voided discount
- **NB_DISCOUNT** - „k”: non-business policy discount
- **VOID_NB_DISCOUNT** - „ks”: voided non-business policy discount



- **SURCHARGE** - „f”: surcharge
- **VOID_SURCHARGE** - „fs”: voided surcharge
- **EMPTIES** - „g”: returnable packaging refund
- **VOID_EMPTIES** - „gs”: voided returnable packaging refund

Response data structure in case of successful item addition

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "docTotal": "1898.00",  
    "documentData": [  
        {  
            "@type": "receiptItem",  
            ...  
        }  
    ]  
}
```

Data structure field explanation:

- **fiscalDayNo** - The serial number of the currently open fiscal day
- **documentId** - The identifier of the opened simplified invoice
- **docTotal** - The current total amount of the simplified invoice
- **documentData** - The data required to display the receipt image

12.9.12.3 Closing the document

Once the items have been added to the opened simplified invoice, the document must be closed. When closing the document, payment information must be provided, which the FAM verifies and stores in the database. After closing the simplified invoice, adding further items to the document is not allowed.

The termination of the document creation by the operator also occurs at this endpoint

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-close

Endpoint Request Objects: DocCloseSimpleInvoice - Closing the simplified invoice in case of a successful sale, DocCloseInterruption - Closing the simplified invoice in case of an interrupted sale (both are descendants of DocCloseRequest)

Endpoint Response Objects: DocCloseResponse

Request data structure in case of successful sales

```
{  
    "@type": "closeSimpleInvoice",  
    "systemId": "{{systemId}}",  
    "documentId": {{documentId}},  
    "docCustomInfo": [  
        {  
            "@type": "text",  
            ...  
        }  
    ]  
}
```



```
        "key": "K00701_THX",
        "description": "Leírás",
        "text": "Köszönjük a vásárlást!",
        "alignment": "CENTER",
        "orderId": 1
    }
],
"paymentDetails": [
    {
        "name": "Készpénz",
        "moneyCat": "CASH",
        "moneySubCat": null,
        "moneyAmount": "5000",
        "currency": "HUF"
    }
],
"serviceFee": null,
"attachment": ...
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=closeSimpleInvoice)
- The rest of the fields are the same as those defined in the [Receipt closure](#) section.

Request data structure in case of interrupted sales

```
{
    "@type": "closeInterruption",
    "systemId": "{{systemId}}",
    "documentDescriptor": {
        "type": "SIMPLE_INVOICE",
        "docId": {{documentId}},
        "fiscalDayNo": {{fiscalDayNo}},
        "interrupted": true
    }
}
```

Data structure field explanation:

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=closeInterruption)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentDescriptor*** - The data structure identifying the documents
 - **type*** = SIMPLE_INVOICE
 - **docId*** - Unique identifier of the simplified invoice
 - **fiscalDayNo*** - Serial number of the fiscal day
 - **interrupted*** - A flag indicating the interruption of the document

The data required to display the receipt image

```
{
    "resultCode": "SUCCESS",
    "resultDesc": null,
    "remainingSum": "0.00",
```



```
"documentData": [
    {
        "@type": "docClose",
        ...
    }
]
"docDownloadInfo": "{...}"
}
```

Data structure field explanation:

- **remainingSum** - Remaining amount to be paid, always „0.00” in the case of exact payment
- **documentData** - The data required to display the receipt image
- **docDownloadInfo** – The content of the QR code to be printed on the receipt copy for the customer, containing the e-receipt download information. The content is explained in the section on [Formation of the output QR code for the E-cash register](#) section.

Response data structure in case of successful receipt interruption

```
{
    "documentData": [
        {
            "@type": "receiptItem",
            ...
        },
        ...
        {
            "@type": "docClose",
            ...
        }
    ],
    "resultCode": "SUCCESS",
    "resultDesc": null,
    "remainingSum": null
}
```

Data structure field explanation

- **documentData** - The data required to display the receipt image. Its content includes all previously added items with opposite signs, as well as the closing data structure.
- The value of all other fields in the response is *null*.

12.9.13 **Invoice**

The invoice contains the buyer's data, the payment deadline, the date of performance, the description of the purchased goods or services, the prices and quantities, as well as the total amount. Furthermore, business or other purpose-specific unique information can also be attached to it.

The creation of an invoice receipt takes place in the following steps:

- Creating the invoice
- Adding any number of items to the receipt



- Closing the receipt by providing payment data

These steps can be submitted to the FAM in several consecutive API calls.

12.9.13.1 Creating an invoice

The creation of an invoice is also only possible within an open fiscal day. Creating the receipt includes saving the invoice data structure into the database, saving the buyer's data into the database, as well as recording the invoice identifiers within the fiscal day. Creating an invoice is almost identical to handling a receipt or a simplified invoice, the differences being the provision of invoice-specific data in the invoice HTTP requests.

The buyer may decide whether they request a paper-based invoice. In this case, the billTo.invoiceType field must be set to PAPER when submitted.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateInvoice (descendant of DocCreateRequest)

Endpoint Response Objects: DocCreateResponse

Request data structure

```
{  
    "@type": "createInvoice",  
    "systemId": "{{systemId}}",  
    "fulfillmentDate": 1720519280905,  
    "paymentDue": 1720519280905,  
    "paymentType": "CASH",  
    "billTo": {  
        "name": "Teszt Elek",  
        "address": {  
            "addressType": "SIMPLE",  
            "countryCode": "HU",  
            "postCode": "1000",  
            "city": "Budapest",  
            "additionalAddressDetail": "Vas utca 33"  
        },  
        "taxNumber": {  
            "taxpayerId": "30000003",  
            "vatCode": "3",  
            "countyCode": "33"  
        },  
        "groupMemberTaxNumber": null,  
        "communityTaxNumber": null,  
        "thirdCountryTaxNumber": null,  
        "customerVatStatus": "DOMESTIC",  
        "invoiceType": "ELECTRONIC",  
        "bankAccountNo": "123123123123123123"  
    },  
    "receiptItems": [{  
        "itemName": "Masszázs",  
        "itemArticleNo": "5998765676545",  
        "itemUnitPrice": "10000",  
        "itemQty": "1.0000",  
    }]
```



```
"itemUnit": "HOUR",
"itemCat": "SALE",
"itemDept": "A",
"itemCustomInfo": [
    {
        "@type": "text",
        "key": "M34001_THAI",
        "description": "Leírás",
        "text": "comment",
        "alignment": "CENTER",
        "orderId": 1
    }
]
}]
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=createInvoice)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **fulfillmentDate*** – Date of performance, if it is submitted with a null value, it inherits the date of the fiscal day.
- **paymentDue*** – Payment deadline, if it is submitted with a null value, it inherits the date of the fiscal day.
- **paymentType*** – Payment method
 - **CASH** – cash payment
 - **WIRE_TRANSFER** – wire transfer
- **billTo*** - Customer details, the data structure is described in the [billTo](#) section.

Response data structure

```
{
    "resultCode": "SUCCESS",
    "resultDesc": null,
    "fiscalDayNo": 1,
    "documentId": 1,
    "docTotalNet": "10000.00",
    "docTotal": "12700.00",
    "docTax": "2700.00",
    "documentData": [
        {
            "@type": "docCreate",
        },
        {
            "@type": "receiptItem"
        }
    ]
}
```



Data structure field explanation:

- **fiscalDayNo** - Serial number of the currently open fiscal day
- **documentId** - Unique identifier of the created invoice
- **docTotalNet** - The current net total of the receipt
- **docTotal** – The actual gross value of the invoice
- **docTax** – The actual VAT value of the invoice
- **documentData** - The data required to display the receipt image

12.9.13.2 Adding an item

Any number of items can be added to the open invoice. When adding items, the FAM saves the item data structure into the database, performs the appropriate arithmetic calculations, and increments the internal counters of the invoice.

API endpoint group details: FAM interface/Document - Receipt handling

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/add-item

Endpoint Request Objects: AddInvoiceItemRequest - in the case of adding an item (descendants of AddItemRequest)

Endpoint Response Objects: AddItemDocumentResponse

Request data structure in the case of adding an item.

```
{  
    "@type": "addInvoiceItem",  
    "systemId": "{{systemId}}",  
    "documentId": {{documentId}},  
    "receiptItems": [  
        {  
            "itemName": "Masszázs olaj",  
            "itemArticleNo": "5998765676545",  
            "itemUnitPrice": "1000",  
            "itemQty": "1.0000",  
            "itemUnit": "PIECE",  
            "itemCat": "SALE",  
            "itemDept": "A",  
            "itemCustomInfo": [  
                {  
                    "@type": "text",  
                    "key": "M60001_VEGAN",  
                    "description": "Leírás",  
                    "text": "comment",  
                    "alignment": "CENTER",  
                    "orderId": 1  
                }  
            ]  
        }  
    ]  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=addInvoiceItem)



- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentId*** - Invoice identifier
- **receiptItems*** - The data structure of the item added to the invoice. The description of the related data structure can be found under the [receiptItems](#) section

The possible values of the given item type in the case of an invoice - itemCat

- **SALE** - „n”: sale
- **VOID_SALE** - „ns” sale cancellation
- **DISCOUNT** - „e”: discount
- **VOID_DISCOUNT** - „es”: cancellation of discount
- **NB_DISCOUNT** - „k”: non-commercial discount
- **VOID_NB_DISCOUNT** - „k” cancellation of non-commercial discount
- **SURCHARGE** - „f”: surcharge
- **VOID_SURCHARGE** - „fs”: cancellation of surcharge
- **EMPTIES** - „g” return of empties
- **VOID_EMPTIES** - „gs”: return of empties cancellation

Response data structure upon successful item addition

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "docTotalNet": "10000.00",  
    "docTotal": "13970.00",  
    "docTax": "27002970.00",  
    "documentData": [  
        {  
            "@type": "receiptItem",  
            ...  
        }  
    ]  
}
```

Data structure field explanation:

- **fiscalDayNo** - Serial number of the currently open fiscal day
- **documentId** - Unique identifier of the created invoice
- **docTotalNet** - The current net total of the receipt
- **docTotal** - The actual gross value of the invoice
- **docTax** - The actual VAT value of the invoice
- **documentData** - The data required to display the receipt image

12.9.13.3 Closing the invoice

If the items have been recorded for the opened invoice, the receipt must be closed. When closing the receipt, it is mandatory to provide payment information, the correctness of which is checked by the FAM and saved into the database. After closing the invoice, adding further items to the receipt is not permitted.



The interruption of receipt creation by the operator also takes place at this endpoint. The invoice can be closed with two types of financial transactions, the use of other payment methods is not allowed:

- Cash payment (moneyCat=CASH)
- Bank transfer (moneyCat=WIRE_TRANSFER)

API endpoint group details: FAM interface/Document - Receipt handling

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-close

Endpoint Request Objects: DocCloseInvoice - in the case of successful invoice closure (descendants of AddItemRInvoice closing in case of successful sale, DocCloseInterruption - invoice closing in case of interrupted sale (descendants of DocCloseRequest))

Endpoint Response Objects: DocCloseResponse

Request data structure in the case of successful document closure

```
{  
    "@type": "closeInvoice",  
    "systemId": "{{systemId}}",  
    "documentId": {{documentId}},  
    "docCustomInfo": [  
        {  
            "@type": "text",  
            "key": "K00005_DANGO",  
            "description": "Leírás",  
            "text": "Köszönjük a vásárlást!",  
            "alignment": "CENTER",  
            "orderId": 1  
        }  
    ],  
    "paymentDetails": [  
        {  
            "name": "Átutalás",  
            "moneyCat": "WIRE_TRANSFER",  
            "moneySubCat": null,  
            "moneyAmount": "5000",  
            "currency": "HUF"  
        }  
    ],  
    "serviceFee": null,  
    "attachment": ...  
}
```

Data structure field explanation

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=closeInterruption)
- **systemId*** - The unique identifier of the FAM instance (AP number)
- **documentDescriptor*** - Data structure identifying the receipts
 - **type*** = INVOICE
 - **docId*** - Invoice identifier
 - **fiscalDayNo*** - Serial number of the fiscal day



- **interrupted*** - Switch indicating the fact of the receipt interruption

Response data structure in case of successful document closure

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "remainingSum": "0.00",  
    "documentData": [  
        {  
            "@type": "docClose",  
            ...  
        }  
    ],  
    "docDownloadInfo": "{...}"  
}
```

Data structure field explanation

- **remainingSum** - Remaining amount payable, in the case of exact payment its value is always "0.00"
- **documentData** - The data required to display the receipt image
- **docDownloadInfo** - The content of the QR code containing the e-receipt download information to be printed on the receipt copy for the customer. Its content is explained in the chapter Formation of the e-cash register output QR code

Response data structure in case of interrupted document closure

```
{  
    "documentData": [  
        {  
            "@type": "receiptItem",  
            ...  
        },  
        ...  
        {  
            "@type": "docClose",  
            ...  
        }  
    ],  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "remainingSum": null  
}
```

Data structure field explanation

- **documentData** - The data required to display the receipt image. Its content is all previously added items with opposite signs, as well as the closing data structure.
- The value of the other fields in the response is null



12.9.14 Custom document

The use of custom documents allows for the printing of coupons, customer information sheets (e.g., shoe care recommendations), pharmacy printouts regarding the preparation of prescribed medicines, and other business or non-business-related forms.

The creation of a Custom Document follows similar steps to sales receipts:

- Creating a custom document
- Adding any number of custom information items to the document
- Closing the document

These steps can be transmitted to the FAM in multiple consecutive API calls

12.9.14.1 Creating a custom document

The creation of a custom document is only possible within an open fiscal day. The creation process includes saving the data structure of the custom document to the database and registering its identifiers within the fiscal day. The data structure of the custom document also allows the inclusion of document items in the HTTP request.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-create

Endpoint Request Objects: DocCreateCustomDoc (descendant of DocCreateRequest)

Endpoint Response Objects: DocCreateDocumentResponse

Request data structure

```
{  
    "@type": "createCustomDoc",  
    "systemId": "{{systemId}}",  
    "customItems": [  
        {  
            "@type": "text",  
            "text": "comment",  
            "alignment": "CENTER",  
            "orderId": 1  
        }  
    ]  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type***- Informs the FAM Document interface about the type of data structure sent in the HTTP request (=createCustomDoc)
- **systemId***- The unique identifier of the FAM instance (AP number)
- **customItems**- The structural composition of the custom information item. Detailed in the "[Adding an Item](#)" subsection.

Response Data Structure (Upon Successful Execution)



```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "fiscalDayNo": 1,  
    "documentId": 1,  
  
    "documentData": [  
        {  
            "@type": "docCreate",  
            ...  
        },  
        {  
            "@type": "text",  
            ...  
        }  
    ]  
}
```

Data structure field explanation:

- **fiscalDayNo**- The serial number of the currently open fiscal day
- **documentId**- The unique identifier of the created custom document.
- **documentData** - The data required to display the receipt image

12.9.14.2 Adding an item

Any number of custom informational items can be registered to open custom documents. During item addition, the FAM stores the item's data structure in its database. Six types of items can be added to custom documents:

- Image (from the FAM file storage)
- Text information
- Bardoce 1D
- Barcode DataMatrix
- Barcode PDF417
- QR code

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/add-item

Endpoint Request Objects: AddCustomItemRequest (descendants of AddItemRequest)

Endpoint Response Objects: AddItemDocumentResponse

Request data structure in case of adding an item

```
{  
    "@type": "addCustomItem",  
    "systemId": "{{systemId}}",  
    "documentId": {{documentId}},  
    "customItems": [{  
        "@type": "text",  
        "text": "comment 1",  
        "alignment": "CENTER",  
        "orderId": 1  
    }]  
}
```



Data structure field explanation:

* Fields marked with * are mandatory

- **@type***- The FAM Document interface informs about the type of data structure sent in the HTTP request (=addCustomItem)
- **systemId***- The unique identifier of the FAM instance (AP number)
- **documentId***- The unique identifier of the custom document
- **customItems***- The data structure of the informational items added to the custom document

Response data structure in case of successful item addition

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "fiscalDayNo": 1,  
    "documentId": 1,  
    "documentData": [  
        {  
            "@type": "text",  
            "orderId": 2,  
            "text": "comment 2",  
            "alignment": "CENTER"  
        }  
    ],  
}
```

Data structure field explanation:

- **fiscalDayNo**- The serial number of the currently open fiscal day
- **documentId**- The unique identifier of the newly opened custom document
- **documentData** - The data required to display the receipt image

12.9.14.3 Document closure

Once the informational items have been added to the opened custom document, the document must be closed. After closing the custom document, adding further items to it is not allowed.

API endpoint group details: FCU interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/doc-close

Endpoint Request Objects: DocCloseCustomDoc – Custom document closure

Endpoint Response Objects: DocCloseResponse

Request data structure in case of successful sales

```
{  
    "@type": "closeCustomDoc",  
    "systemId": "{{systemId}}",  
    "documentId": {{documentId}},  
    "customItems": [  
        {  
            "@type": "text",  
            "text": "Köszönjük a vásárlást!",  
            "alignment": "CENTER",  
            "orderId": 3  
        },  
        {  
            "@type": "text",  
            "text": "VÁROSI KÖLCSÖNÖLÉS",  
            "alignment": "CENTER",  
            "orderId": 4  
        }  
    ]  
}
```



```
        "@type": "text",
        "text": "De tényleg!",
        "alignment": "CENTER",
        "orderId": 4
    }
],
"attachment": ...
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type***- Informs the FAM Document interface about the type of data structure sent in the HTTP request (=closeCustomDoc)
- **systemId***- The unique identifier of the FAM instance (AP number)
- **documentId***- The unique identifier of the custom document
- **customItems**- Additional unique informational items related to the custom document
- **attachment** - The attachment of the custom document

Response data structure in case of successful closure

```
{
    "resultCode": "SUCCESS",
    "resultDesc": null,
    "documentData": [
        {
            "@type": "docClose",
            ...
        }
    ]
}
```

Data structure field explanation

- **documentData** - The data required to display the receipt image

Response data structure in case of interruption

```
{
    "documentData": [
        {
            "@type": "text",
            ...
        },
        ...
        {
            "@type": "docClose",
            ...
        }
    ],
    "resultCode": "SUCCESS",
    "resultDesc": null,
    "remainingSum": null
}
```

Data structure field explanation:

- **resultCode**- The identifier code of the task result



- **resultDesc**- A short description of the task result
- **documentData** - The data required to display the receipt image

The data structure does not change compared to a failed request.

The system performs general request validations ([General request validations and response messages](#)). There are no endpoint-specific checks.

12.9.15 Data transfer from customer application

If the e-cash register reads customer data from the customer application, only the encryption key and date need to be transferred to the FAM.

- The FAM accepts the data only if a fiscal day is open and there is an open sales receipt. If no sales receipt (receipt, simplified invoice, void receipt, or modification receipt) is open at the time of reading, a new document must first be opened in the FAM before the customer data can be transferred.
- If a receipt is open, the data can be transferred at any time before the docClose call.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/submit-key

Endpoint Request Objects: SubmitKeyRequest

Endpoint Response Objects: SubmitKeyResponse

Request data structure

```
{  
    "systemId": "{{systemId}}",  
    "documentDescriptor": {  
        "type": "RECEIPT",  
        "fiscalDayNo": {{fiscalDayNo}},  
        "docId": {{documentId}}  
    },  
    "publicKey": "...",  
    "date": 1715088749000,  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **systemId***- The unique identifier of the FAM instance (AP number)
- **documentDescriptor***- The data structure identifying the receipts, its structure is detailed in the [documentDescriptor](#) section.
- **publicKey*** – The encryption key read from the customer application
- **date*** - The UTC date read from the customer application (search key timestamp) converted into Unix time format with millisecond resolution

Response Data Structure (Upon Successful Execution)

```
{
```



```
"resultCode": "SUCCESS",
"resultDesc": null
}
```

12.9.16 Modification of the open receipt type

The FAM provides the option to change the type of already open receipt, which is useful in the following cases:

- The operator of the e-cash register starts adding items to the receipt, and the customer then requests an invoice. By providing billing information, the open receipt type changes to a simplified invoice.
- The operator of the e-cash register begins preparing a simplified invoice at the customer's request, but the customer changes their mind and decides they only need a receipt.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/change-doc-type

Endpoint Request Objects:

- ChangeDocTypeToSimpleInvoiceRequest (receipt → simple invoice)
- ChangeDocTypeToReceiptRequest (simple invoice → receipt)

Endpoint Response Objects: ChangeDocTypeResponse

Request data structure (receipt → simple invoice)

```
{
    "@type": "changeToSimpleInvoice",
    "systemId": "{{systemId}}",
    "fiscalDayNo": {{fiscalDayNo}},
    "docId": {{documentId}},
    "billTo": {
        "name": "Teszt Elek",
        "address": {
            "addressType": "SIMPLE",
            "countryCode": "HU",
            "postCode": "1000",
            "city": "Budapest",
            "additionalAddressDetail": "Vas utca 33"
        },
        "taxNumber": {
            "taxpayerId": "30000003",
            "vatCode": "3",
            "countyCode": "33"
        },
        "groupMemberTaxNumber": null,
        "communityTaxNumber": null,
        "thirdCountryTaxNumber": null,
        "customerVatStatus": "DOMESTIC",
        "invoiceType": "ELECTRONIC",
        "bankAccountNo": "123123123123123123"
    },
}
```



Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=changeToSimpleInvoice)
- **systemId***- The unique identifier of the FAM instance (AP number)
- **fiscalDayNo***- Serial number of the fiscal day
- **docId***- Unique document identifier
- **billTo*** - Customer data, its data structure is detailed in the [billTo](#)- Customer Data section

Request data structure (simplified invoice à receipt)

```
{  
    "@type": "changeToReceipt",  
    "systemId": "{{systemId}}",  
    "fiscalDayNo": {{fiscalDayNo}},  
    "docId": {{documentId}},  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **@type*** - Informs the FAM Document interface about the type of data structure sent in the HTTP request (=changeToSimpleInvoice)
- **systemId***- The unique identifier of the FAM instance (AP number)
- **fiscalDayNo***- Serial number of the fiscal day
- **docId***-The document identifier

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,  
    "documentDescriptor": {  
        "type": "SIMPLE_INVOICE",  
        "fiscalDayNo": 99,  
        "docId": 41  
    },  
}
```

Data structure field explanation:

- **documentDescriptor***- The data structure identifying the documents
 - **type***- The **new** type of the document
 - **fiscalDayNo***- Serial number of the fiscal day
 - **docId*** - Unique document identifier

Validation/Description	Result Code	J	Action Required
A system error occurred while changing the document type. The document type cannot be changed.	CANNOT_CHANGE_DOCUMENT_TYPE	T	Resend the request data structure.

12.9.17 Verification of payment methods

The FAM provides the possibility for the application to check the total of the means of payment provided by the customer before closing the sales receipt. The use of this function is



optional, its operation is the same as the verification process of paymentDetails when closing the receipt, and its return value is also the same, the only difference being that it does not close the receipt.

It can be used, for example, in cases when the customer uses several means of payment. First, the seller enters the amount of cash provided into the application, the application sends the means of payment provided so far to the FAM, and the FAM returns how much of the remaining amount the customer still must pay. This can then be automatically sent, for example, to the bank card terminal.

By submitting the paymentDetails object to the endpoint, the application receives back whether the assembled payment “cart” covers the total amount of the receipt, and it also returns the remaining amount if not.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: POST

Context Root: /fam/v1

Endpoint URL: /doc/calculate-payments

Endpoint Request Objects: CalculatePaymentsRequest

Endpoint Response Objects: CalculatePaymentsResponse

Request Data Structure

```
{  
    "systemId": "{{systemId}}",  
    "documentDescriptor": {  
        "type": "RECEIPT",  
        "docId": {{documentId}},  
        "fiscalDayNo": {{fiscalDayNo}}  
    }  
    "paymentDetails": [  
        {  
            "name": "Készpénz",  
            "moneyCat": "CASH",  
            "moneySubCat": null,  
            "moneyAmount": "5000",  
            "currency": "HUF"  
        }  
    ]  
}
```

Data structure field explanation:

* Fields marked with * are mandatory

- **systemId***- The unique identifier of the FAM instance (AP number)
- **documentDescriptor** - the descriptor of the receipt
- **paymentDetails*** - The structure of paymentDetails is described in the chapter [“PaymentDetails – Receipt Monetary Data”](#)

Response Data Structure upon successful completion

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": null,
```



```
"paymentDetails": [
    {
        "name": "Készpénz",
        "moneyCat": "CASH",
        "moneySubCat": null,
        "moneyAmount": "5000",
        "currency": "HUF"
    }
]
"remainingSum": "0.00",
}
```

Data structure field explanation:

- **paymentDetails*** - The structure of paymentDetails is described in the chapter "[PaymentDetails – Receipt Monetary Data](#)"
- **remainingSum** - Remaining amount payable, in the case of exact payment its value is always "0.00"

The system performs the general request validations. There is no specific validation for the endpoint.

12.10 Querying the document list based on a time interval, optionally by type

Returns a list of documents issued within the specified period and optionally filtered by document type, along with their key parameters.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /doc/list/{systemId}/{startTime}/{endTime}?documentType={...}

Endpoint Response Objects: DocumentListResponse

Query Parameters in the URL:

* Fields marked with * are mandatory

- **systemId*** - The unique identifier of the FAM instance (AP number)
- **startTime*** - Start of the query period
- **endTime*** - End of the query period
- **documentType** - Type of documents to be queried
Possible values are detailed in the "[Document Type](#)" section.

Response Data Structure (Upon Successful Execution):

```
{
    "resultCode": "SUCCESS",
    "resultDesc": "",
    "documents": [
        {
            "type": RECEIPT,
            "fiscalDayNo": 1,
            "docId": 1,
            "interrupted": false,
            "docTotal": "1000.00",
        }
    ]
}
```



```
        "docCreationDate": 1727203335672,  
        "docNo": "NY-Y1950001/20000002/0001/00001"  
    },  
    ...  
}
```

Data structure field explanation:

- **documents** - A list used to identify the receipt and the data structure of its individual elements
 - **type** - Type of the receipt
 - **fiscalDayNo** - Serial number of the fiscal day
 - **docId** - Unique identifier of the receipt
 - **interrupted** - Indicator showing whether the receipt was interrupted
 - **docTotal** – Total amount of the receipt, returned by FAM only for applicable receipt types
 - **docCreationDate** – Date of receipt creation
 - **docNo** – Serial number of the receipt

12.11 Retrieving the data structure used for receipt image generation

Returns data for any receipt to generate a receipt image. This function allows the numerical values to be returned in raw input format or country-specific formatting.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: GET

Context Root: /fam/v1

Endpoint

URL:

/doc/document/{systemId}/{fiscalDayNo}/{documentType}/{documentId}/{formatType}

Endpoint Response Objects: GetDocumentDataResponse

Query Parameters in the URL:

* Fields marked with * are mandatory

- **systemId*** - The unique identifier of the FAM instance (AP number)
- **fiscalDayNo*** - Serial number of the fiscal day
- **documentType*** - Type of the receipts to be queried
The possible values are detailed in the [Document Type](#) section.
- **documentId*** - Unique identifier of the receipt
- **formatType*** - Formatting type
Value options:
 - RAW – Data structure containing the receipt details in an unformatted way, suitable for machine processing (e.g., for creating a void receipt).
 - FORMATTED – The requested data structure with numerical values formatted for printing (thousands separator, decimal comma), suitable for receipt copy printing.

Response Data Structure (Upon Successful Execution)

{



```
"resultCode": "SUCCESS",
"resultDesc": "",
"documentData": [...]
"docDownloadInfo": [...]
}
```

Data structure field explanation:

- **documentData** - The receipt image data: Explained in the Receipt Image Data section.
- **docDownloadInfo** – The content of the QR code to be printed on the receipt copy.

DocumentData structure in RAW formatting type:

```
"documentData": [
  {
    "@type": "docCreate",
    ...
  },
  {
    "@type": "receiptItem",
    "itemId": 1,
    "itemRef": null,
    "itemName": "Cherry tomato",
    "itemArticleNo": "5998765676545",
    "itemUnitPrice": "1499.00",
    "itemQty": "1.0000",
    "itemUnit": "KILOGRAM",
    "itemSubUnit": null,
    "itemCat": "SALE",
    "itemDept": "A",
    "itemSum": "1499.00",
    "itemCustomInfo": [
      {
        "@type": "text",
        "key": "T80001_FURTOS",
        "description": "Leírás",
        "orderId": 1,
        "text": "comment",
        "alignment": "CENTER"
      }
    ]
  },
  {
    "@type": "docClose",
    "systemId": "Q00000001",
    "footNote": null,
    "docCloseDate": "2024.09.30 14:07:13",
    "docValidationCode": {
      "text": "NAV ELLENŐRZŐ KÓD",
      "value": "D8164"
    },
    "docNo": {
      "text": "BIZONYLATSZÁM",
      "value": "NY-Q00000001/20000002/0001/00001"
    },
    "docInterruption": false,
    "docInterruptionCause": null,
    "docTotal": {
      "text": "ÖSSZESEN",
      "value": "1898"
    }
  }
]
```



```
},
"round": "-2",
"docCustomInfo": [
{
    "@type": "text",
    "key": "K60701_KOSZON",
    "description": "Leírás",
    "orderId": 1,
    "text": "Köszönjük a vásárlást!",
    "alignment": "CENTER"
}
],
"paymentDetails": [
{
    "name": "KÉSZPÉNZ",
    "moneyCat": "CASH",
    "moneySubCat": null,
    "moneyAmount": "5000.00",
    "moneyLocalValue": "5000.00",
    "currency": "HUF",
    "currencyXchRate": "1.00",
    "currencySymbol": "Ft",
    "isLocalCurrency": true
},
{
    "name": "VISSZAJÁRÓ",
    "moneyCat": "CHANGE",
    "moneySubCat": null,
    "moneyAmount": "-3100.00",
    "moneyLocalValue": "-3100.00",
    "currency": "HUF",
    "currencyXchRate": "1.00",
    "currencySymbol": "Ft",
    "isLocalCurrency": true
},
{
    "name": "KEREKÍTÉS",
    "moneyCat": "ROUND",
    "moneySubCat": null,
    "moneyAmount": "-2.00",
    "moneyLocalValue": "-2.00",
    "currency": "HUF",
    "currencyXchRate": "1.00",
    "currencySymbol": "Ft",
    "isLocalCurrency": true
}
]
}
```

DocumentData structure in FORMATTED formatting type:

```
"documentData": [
{
    "@type": "docCreate",
    ...
},
{
    "@type": "receiptItem",
    ...
}
]
```



```
"itemId": 1,
"itemRef": null,
"itemName": "Cherry tomato",
"itemArticleNo": "5998765676545",
"itemUnitPrice": "1 499",
"itemQty": "1",
"itemUnit": "KILOGRAM",
"itemSubUnit": null,
"itemCat": "SALE",
"itemDept": "A",
"itemSum": "1 499",
"itemCustomInfo": [
    {
        "@type": "text",
        "key": "T80001_FURTOS",
        "description": "Leírás",
        "orderId": 1,
        "text": "comment",
        "alignment": "CENTER"
    }
],
{
    "@type": "docClose",
    "systemId": "Q00000001",
    "footNote": null,
    "docCloseDate": "2024.09.30 14:07:13",
    "docValidationCode": {
        "text": "NAV ELLENŐRZŐ KÓD",
        "value": "D8164"
    },
    "docNo": {
        "text": "BIZONYLATSZÁM",
        "value": "NY-Q00000001/20000002/0001/00001"
    },
    "docInterruption": false,
    "docInterruptionCause": null,
    "docTotal": {
        "text": "ÖSSZESEN",
        "value": "1 898"
    },
    "round": "-2",
    "docCustomInfo": [
        {
            "@type": "text",
            "key": "K90090_DANKE",
            "description": "Leírás",
            "orderId": 1,
            "text": "Köszönjük a vásárlást!",
            "alignment": "CENTER"
        }
    ],
    "paymentDetails": [
        {
            "name": "KÉSZPÉNZ",
            "moneyCat": "CASH",
            "moneySubCat": null,
            "moneyAmount": "5 000",
            "order": 1
        }
    ]
}
```



```
        "moneyLocalValue": "5 000",
        "currency": "HUF",
        "currencyXchRate": "1",
        "currencySymbol": "Ft",
        "isLocalCurrency": true
    },
    {
        "name": "VISSZAJÁRÓ",
        "moneyCat": "CHANGE",
        "moneySubCat": null,
        "moneyAmount": "-3 100",
        "moneyLocalValue": "-3 100",
        "currency": "HUF",
        "currencyXchRate": "1",
        "currencySymbol": "Ft",
        "isLocalCurrency": true
    },
    {
        "name": "KEREKÍTÉS",
        "moneyCat": "ROUND",
        "moneySubCat": null,
        "moneyAmount": "-2",
        "moneyLocalValue": "-2",
        "currency": "HUF",
        "currencyXchRate": "1",
        "currencySymbol": "Ft",
        "isLocalCurrency": true
    }
]
}
```

12.12 Requesting a complete receipt image

The FAM system supports the generation of a complete, print-ready receipt image to simplify client-side printing tasks. The use of this function is **optional**, as the receipt image can also be assembled on the client side.

The endpoint can return the receipt image in PNG (compressed bitmap) and HTML formats, with HTML primarily intended for on-screen display.

To generate the receipt image, the printer settings described in the [Peripheral settings](#) for the client-connected peripheral must be configured as follows:

- **sys.printer.selected** – Value: "client default" (the client-side default printer.)
- **sys.client.printer.default.dpi** – The print head resolution in DPI, its most common value is 203, it can be read from the technical datasheet of the specific printer.
- **sys.client.printer.default.imageWidthPx** – The printer's print width in pixels read from the technical datasheet of the printer.
 - For **57 mm (2") paper**, the typical width is **384 pixels**.
 - For **80 mm (3") paper**, the typical width is **576 pixels**.
- **sys.client.printer.default.paperWidthChr** – the number of characters displayed in one line of the receipt, given as an integer
 - In the case of using 57 mm (2") and 80 mm (3") paper, the factory default value is 48 characters.



If FAM generates an output QR code for the receipt, it will also be included in the returned image.

API endpoint group details: FAM interface/Document - Document Management

Endpoint Component: DocumentController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /doc/document-

image/{systemId}/{fiscalDayNo}/{documentType}/{documentId}/{formatType}

Endpoint Response Objects: GetDocumentDataResponse

Query parameters in the URL:

* Fields marked with * are mandatory

- **systemId*** - The unique identifier of the FAM instance (AP number)
- **fiscalDayNo*** - Serial number of the fiscal day
- **documentType*** - Type of document to be queried
Possible values are provided in the [Document Type](#) section.
- **documentId*** - Identifier of the document
- **formatType*** – The response format type for the receipt image Possible values for format type:
 - **PNG** – 1-bit black-and-white PNG format image
 - **HTML**

Response Data Structure (Upon Successful Execution)

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "data": [...]  
}
```

Data structure field explanation:

- **data** - Receipt Image in Base64 Encoding

12.13 File management

The e-cash register signals the successful submission of the e-receipt to the Receipt Archive (Nyugtatár) by displaying a graphical element (animated GIF) and playing a sound notification (WAV file). These files are provided and made available for download by the Hungarian Tax Authority (NAV) for use by the Fiscal Unit (Adóügyi Egység) of the e-cash register.

Both files are available for download from the FAM file interface in a single ZIP archive. The need for downloading updated files is determined through status synchronization. The FAM instance status descriptor contains the ZIP file identifier. If this identifier differs from the one stored in the e-cash register, the system must download the new ZIP file and extract its contents for use.

This mechanism ensures that the latest version of the required audio-visual notifications is always available in the e-cash register.

API endpoint group details: FAM interface/File



Endpoint Component: FileController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /file/{systemId}/{fileType}/{ fileId}

Endpoint Response Objects: FileResponse

Query parameters in the URL:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)
- **fileType*** – The type of file to be downloaded. In the case of a ZIP archive, this must be set to „MEDIA_PACKAGE”
- **fileId*** – The unique identifier of the data file, which is provided in the navMedia attribute of the complete status description.

Response Data Structure (Upon Successful Execution):

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "data": "iVBORw0KGgoAAAANSUhE...AAACDi5D8LWz2gPzhA3wAAAABJRU5ErkJggg=="  
}
```

Data structure field explanation:

- **data** – The ZIP file containing the image and audio file is base64 encoded

12.14 Auditor identification

In compliance with the "Generating the Signature Verification QR Code for the E-Cash Register" section, the FAM system provides the capability to retrieve an officially signed QR code that contains characteristic data specific to the ePG (electronic cash register).

This QR code allows auditors and regulatory authorities to verify the authenticity and integrity of the e-cash register system by scanning and decoding the signed information embedded in the QR code.

API endpoint group details: FAM interface/File

Endpoint Component: SystemController

HTTP Method: GET

Context Root: /fam/v1

Endpoint URL: /system/auditor-info/{systemId}

Endpoint Response Objects: AuditorInfoResponse

Query Parameters in the URL:

* Fields marked with * are mandatory

- **systemId*** – The unique identifier of the FAM instance (AP number)

Response Data Structure (Upon Successful Execution):

```
{  
    "resultCode": "SUCCESS",  
    "resultDesc": "",  
    "data":  
"3|DZRqtdA==|AB99912345|N98765432|E11110123456789|S11119876543210|Gi9LMP5ZoNRhw
```



a/pJPEGDh1Q1g6lY8K4E+0EgM+/0cRps2c1FfykqBW8k+8sUSjBK35D3VM0DmMDeiGBN13s9Ag==|CEMT
ysHwssISqNnBMQi711aQ="
}

Data structure field explanation:

- **data** – The QR code containing the ePG (electronic cash register) data

13 Optional supplementary fields for documents

The data structure of sales receipts and reports allows for the recording of optional data related to a given document. This chapter outlines the rules for using supplementary fields.

13.1 The purpose of the supplementary information element

The data structure of e-cash register receipts provides the possibility to include data not previously defined. This data may belong to individual items of the e-receipt or to the entire e-receipt.

This information serves the purpose of providing data for customer information beyond the mandatory elements of the receipt image previously printed on paper-based receipts, as well as for supplying additional data that go beyond the limitations of paper, in the form of key-value pairs with descriptions. The information element also defines the form in which the data content is displayed (e.g. line or dot code).

Related data can also be handled within one information element, for example, the bank card transaction slip part that was previously printed in several lines on the receipt can be submitted in one element as continuous text in the e-receipt system, with line breaks indicated by the “\n” code, if it fits within the character limit of the field.

It is advisable to make the content of the supplementary information element easy and simple to define. This enables the quick and cost-effective implementation of new business needs related to the receipt data content without requiring an authorization process.

13.2 Sales receipts

For sales receipts, the following supplementary fields can be used:

- Supplementary fields of receipt data (receiptCore, otherDocumentCore, simplifiedInvoiceCore):
 - Document-level supplementary data: up to 100 elements
 - Item-level supplementary data: up to 10 elements per item
- Customer supplementary data (receiptAdditional, otherDocumentAdditional, simplifiedInvoiceAdditional):
 - Document-level supplementary data: up to 10 elements
 - Item-level supplementary data: up to 10 elements per item
 - Attachment: one per receipt, with a maximum file size of 512kB

For customer supplementary data, item-level data must refer to the corresponding item line of the base document to which the data (up to 10 elements per receipt item) belongs. Only

existing receipt items can be referenced with item-level supplementary data. A supplementary field can belong to either a single item or the entire document.

13.3 Reports

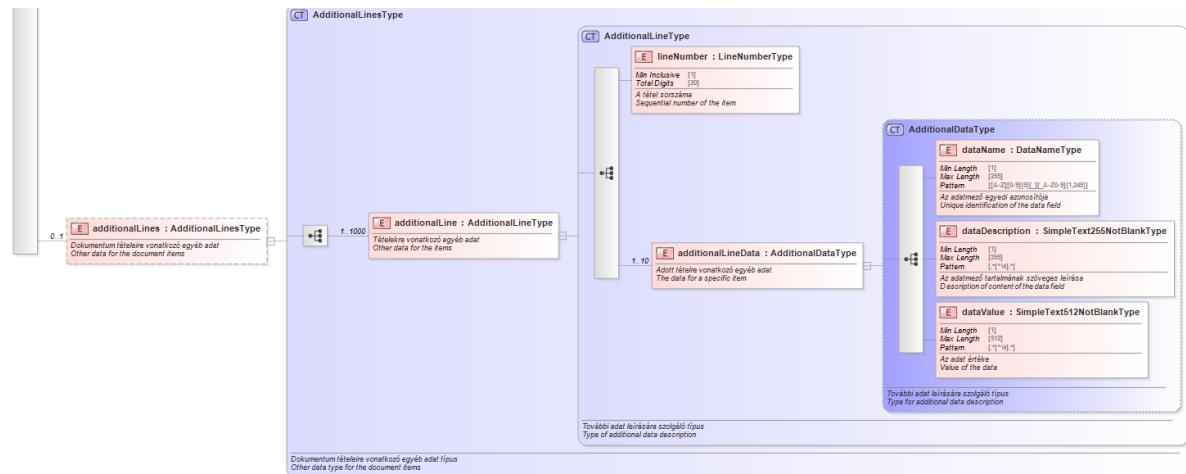
For report-type receipts that can also be issued to customers (certain cash movement receipts and other documents), the supplementary fields are as follows:

- Supplementary fields of receipt data (CoreReport):
 - Document-level supplementary data: up to 10 elements
- Customer supplementary data (CustomerReport):
 - Document-level supplementary data: up to 10 elements
 - Item-level supplementary data: up to 10 elements per item
 - Attachment: one per receipt, with a maximum file size of 512kB

The same rules apply to supplementary customer data as for sales receipts.

13.4 Structure of the supplementary information element

The elements attached to a receipt or receipt items are of type AdditionalDataType, as illustrated in the following diagram:



The structure consists of three fields:

- **dataName**: the type identifier of the supplementary information element, up to 255 characters. Its content is defined by the following regular expression:
 - $[A-Z][0-9]{5}[_[_A-Z0-9]{1,249}]$
- **dataDescription**: a textual description of the content of the data field, up to 255 characters
- **dataValue**: the informational element's data content, up to 512 characters

The identifier written in the dataName field must clearly refer to the nature of the data recorded in the data field, which may include an attached warranty certificate, an online-accessible warranty certificate, a downloadable user manual, active ingredient content, country of origin, manufacturing LOT number, energy value (kcal), etc.

The content of dataName serves machine processing and does not need to be displayed in customer applications or on receipt copies.



It must also be indicated in the `dataName` field if the information element contains confidential data that cannot be printed in full on the receipt copy, or the display of which in the customer application is tied to a separate user interaction (confirmation of reveal). Such data may include, for example, the number of a discount card, the submission of which together with receipt data is mandated as compulsory data provision by legislation. In such a case, the type identifier submitted in the `dataName` field must be suffixed with the postfix “`_BIZALMAS`”. Partial display of the confidential data – as confirmation of data recording – is permitted, but for this a separate information element must be used, containing the displayable value, e.g. most of the card number replaced with asterisks, leaving the last 3 digits visible. The communication of confidential data that concerns only the customer must be implemented using supplementary information elements placed in the customer envelope.

The content of the “`dataDescription`” field appears together with the receipt data as the explanation of the given supplementary element, and is also included on the printed copy. Since the field cannot be empty, if there is no need to display a description related to the value (for example, the textual value “Thank you for your purchase” does not need to be explained with a separate description), a single non-breaking space (`U+00A0`) or zero-width space (`U+200B`) character may be entered into the field.

The `dataValue` must contain data corresponding to the `dataName` identifier, for example, the filename of an attached warranty certificate, the URL of the warranty certificate, the URL of the manual, “`10mg/tbl metamizole`,” “Country of origin: Hungary,” “`LOT 12345677`,” “`234 kcal/100g`.”

13.5 Examples to help interpretation

13.5.1 Case A

According to the provisions of Government Decree 726/2020. (XII. 31.) on the determination of procedural rules regarding the provision of an application enabling the permanent deletion of data, in certain cases the merchant provides the customer with the data deletion code necessary for the permanent inaccessibility of durable storage media, and supplies data about the transfer to the NAV. If the merchant wishes to perform the data provision using an e-cash register and the legislation allows for this, an effective solution is to include this data provision in the appropriate supplementary information element of the e-receipt, because this way the customer’s own code may also remain accessible to them.

The access path of the supplementary data: `receiptAdditional` → `additionalLines` → `additionalLine` → `lineAdditionalData`:

Data element name	Data element content
<code>dataName</code>	<code>A10000_VEGLEGES_ADATTORLO_KOD</code>
<code>dataDescription</code>	Permanent data deletion code
<code>dataValue</code>	The given code, for example “ <code>12K4-567P-9123-4C67</code> ”

13.5.2 Case B

The retail chain operating the e-cash register wishes in the future, of its own decision, to indicate on the e-receipt that it was issued on an extra promotional day, and they hope that as many customer applications as possible will highlight this for the customer.



For the quick and efficient implementation of the above business need, the retail chain will ensure that the following elements appear on all such e-receipts among the receipt data (CoreReport) issued on the given days.

The access path of the supplementary data: receiptAdditional → additionalHead → AdditionalData:

Data element name	Data element content
dataName	A90111_AKCIOSNAP
dataDescription	Marking of promotional day on the e-receipt
dataValue	TRUE

At the same time, it contacts the known customer application manufacturers to apply the graphical elements defined by the chain when displaying e-receipts containing this marking.

13.5.3 Case C

Following the launch of the e-cash register system, a new regulation defines a data reporting obligation regarding the pip diameter of dice available in commercial trade, expressed in millimeters. The data provision can also be fulfilled on the receipt issued for their sale.

For the quick and efficient fulfillment of the above obligation, the competent authority announces which dataName element must be included among the receipt data (CoreReport) on the e-cash registers to meet the new obligation, and how fractional numbers must be displayed in the dataValue element. The affected dice specialty shops ensure that the following elements appear for the affected e-receipt items with the dataName element content defined by the authority.

The access path of the supplementary data: receiptAdditional → additionalLines → additionalLine → lineAdditionalData:

Data element name	Data element content
dataName	A00111_DOBOKOCKA_PATM
dataDescription	Dice pip diameter in millimeters
dataValue	The given value, for example "1.2"

At the same time, it contacts the known customer application manufacturers to apply the graphical elements defined by the chain when displaying e-receipts containing this marking.

13.5.4 Case D

Yielding to customer pressure, the stores organized into one retail association decide to give coupons to customers in the case of purchases above a certain amount. They intend to indicate its value expressed in forints uniformly among the customer-specific data of the e-receipt. They trust that the distributors of the customer applications will prepare them for the proper handling of these coupons.



For the quick and efficient implementation of the above business need, the affected stores will ensure that, for the relevant receipts, their data entry device includes the following elements on the receipt among the customer supplementary data (CustomerReport).

The access path of the supplementary data: receiptAdditional → additionalHead → AdditionalData:

Data element name	Data element content
dataName	C00011_NETWORKNAMECOUPON
dataDescription	The value of the coupon of Association X in HUF
dataValue	The given amount, for example "1234"

At the same time, it contacts the known customer application manufacturers to indicate to the customer, using the graphical elements defined by the association, the fact and value of the coupons received when displaying e-receipts containing this marking.

13.6 List of standard supplementary elements

To ensure the uniform use of supplementary elements of the same subject, the NAV maintains and continuously publishes, as part of this Developer Documentation, the list of standard supplementary elements. The current state of the list is as follows:

Serial number:	1
Level	Item
dataName	A10000_VEGLLEGES_ADATTORLO_KOD
dataDescription	Permanent data deletion code
dataValue	Code provided „12K4-567P-9123-4C67”

Serial number:	2
Level	Item
dataName	N00001_GARANCIA
dataDescription	See the use case titled “ Warranty information ”
dataValue	See the use case titled “ Warranty information ”

Serial number:	3
Level	Receipt
dataName	N00002_KUPON
dataDescription	See the use case titled “ Coupon information ”
dataValue	See the use case titled “ Coupon information ”

Serial number:	4
Level	Receipt
dataName	N00003_GYORSKOD
dataDescription	See the use case titled “ Quick access barcode ”
dataValue	See the use case titled “ Quick access barcode ”



Serial number:	5
Level	Receipt or item
dataName	N00004_***
dataDescription	See the use case titled " Line or dot code "
dataValue	See the use case titled " Line or dot code "

Serial number:	6
Level	Receipt or item
dataName	N00005_***
dataDescription	See the use case titled " Image "
dataValue	See the use case titled Image "

Serial number:	7
Level	Receipt or item
dataName	N00006_MELLEKLET
dataDescription	See the use case titled " General file attachment reference "
dataValue	See the use case titled " General file attachment reference "

Proposals for the extension of standard supplementary elements are awaited by the NAV on the public GitHub page of the e-receipt system.

13.7 Document attachment

A single attachment of up to 512kB in size can be added to a document in base64 encoding. This size limit applies to the base64-encoded file.

The structure must include a file extension that clearly and accurately indicates the format of the attachment, such as pdf, jpg, png, or zip.

At least one supplementary information structure (AdditionalDataType) must reference the attached file, with the dataName field clearly specifying the function of the attachment (e.g., warranty certificate).

If a single data file is attached to a document, the filename must be specified in the supplementary information element that references the attachment, e.g., "warranty-1234567.pdf."

If multiple files are attached, they must be placed in a compressed (zip) archive and attached to the document. In this case, the 512kB size limit applies to the entire compressed package. Every file contained within the compressed package must be referenced from a supplementary information element, which may be item-level or document-level.

13.8 Use cases

To ensure that customer applications handle certain optional customer information in a standardized manner, this data must be embedded in the document data structures in the unified format defined here within the general supplementary fields.

The standardized data structure applies to the following supplementary information:

- Warranty information



- Coupons
- Quick access barcodes (e.g. gate-opening code for self-checkout counters)
- Bar codes
- Images

In the cases listed, the name to be written in the dataName field is fixed. It begins with "N", followed by a number characteristic of the given information group, and then a short, written-out description. The dataName designation starting with "Nxxxxx" may only be used for data specified in the Developer Documentation.

The dataDescription field is optional, but for easier identification, it should refer to the content of the given information and help distinguish similar data entries, e.g., "drill warranty certificate" or "hairdryer warranty certificate" on the same sales receipt.

If the requirement specifies, in the case of complex content, the value of dataValue must be filled in the following compact format characteristic of the given information type:

- A UTF-8 encoded character string without line breaks.
- The first data field consists of the first six characters of dataName ("Nxxxxx"), which clearly identifies the type of information package.
- Data fields are separated by the | character.
- If the | character appears within the data, it must be escaped with \ (backslash), i.e., \\|. The escape character itself is also escaped (e.g., \\\|).
- If the data contains a line break, it is replaced with the \n character sequence.
- The first character of each data field (except the first one) serves as the field identifier.
- Only non-empty data fields should be included.
- All data is represented as a string, even if it could also be interpreted as a number.

The data field thus defines the structure, the format printable on the receipt copy, or supports the machine processing of data, which can be interpreted in the same way when read by customer applications (if it contains all necessary data).

The dataValue may also contain continuous text, which can be displayed in multiple lines on the receipt image or in the customer application. The "\n" marker can also be used for line breaking.

13.8.1 Warranty information

The designation for warranty information (dataName field): **N00001_GARANCIA**
The dataValue content may include the following fields as described in the table below:

Field identifier	Data description
F	Filename of the attached warranty document
U	URL link for downloading the individual warranty document
E	Warranty expiration date (YYYY-MM-DD format)
T	Warranty duration with unit of measurement "36 months", "2 years"
I	Additional textual information



Interpretation example 1:

A PDF warranty document is attached to the third receipt item, with a warranty expiration date of November 4, 2026.

CustomerDocumentType Additional Content:

```
<receiptAdditional>
    <documentNumber>NY-T00100001/12345678/0012/00034</documentNumber>
    <attachment>
        <fileBinary>SWR1IGtlcs08bCBhIGbDowpsIGJpbs0hcm1zYQ==</fileBinary>
        <fileExtension>PDF</fileExtension>
    </attachment>
    <additionalLines>
        <additionalLine>
            <lineNumber>3</lineNumber>
            <lineAdditionalData>
                <dataName>N00001_GARANCIA</dataName>
                <dataDescription>Maketo garanciajegy</dataDescription>
                <dataValue>N00001|Fmaketo-warr-serial-123456.pdf|E2026-11-04|
IKészülékregisztráció esetén + 12 hónap garancia ajándékba.\nRészletek a
https://www.maketo.hu/garancia oldalon.</dataValue>
            </lineAdditionalData>
        </additionalLine>
    </additionalLines>
</receiptAdditional>
```

Interpretation example 2:

Two PDF warranty documents are attached to receipt items 2 and 6, bundled into a ZIP archive. A downloadable warranty document is linked to receipt item 9, with warranty periods of 12, 24, and 18 months.

The 2 warranty document – „*makdaralo-garjegy-123.pdf*” and „*bifliator-gar-567.pdf*” – is placed in a common zip file, into the root of the ZIP (without subdirectories).

CustomerDocumentType Additional Content):

```
<receiptAdditional>
    <documentNumber>NY-T00100001/12345678/0056/00078</documentNumber>
    <attachment>
        <fileBinary>
UEsDBBQACAAIAEja1kAAAAAAAAAAAUAZACAAbWFrZGFyYWxvLWdhcmp1Z3ktMTIzLnBkZlVUDQAHV
loyZ1laMmdWWjJndXgLAEE9QEAAAQUAAAAcwhwceMCAFBLBwh3hLlkBwAAAAUAAABQSMEFAIAAgAUa
NrWQAAAAAAAABQAAABUAIBiaWZsaWF0b3ItZ2FyLTU2Ny5wZGZVVA0AB2paMmdsWjJnaloyZ3V4CwA
BBPUBAAAEEFAAAAHM1cHHjAgBQSwcId4S5ZAcAAAAFAAAAEsBAhQDFAAIAAgAR6NrWYeEuWQHAAAABQAA
ABkAIAAAAAAAAKSBAAAAG1ha2RhcmFsby1nYXJqZWd5LTEyMy5wZGZVVA0AB1ZaMmdZwjJnVloyZ
3V4CwABBPUAAAEEFAAAAFBLAQIUAxQACAAIAFGja1l3hLlkBwAAAAUAAAACAAAAAAACkgW4AAA
BiaWZsaWF0b3ItZ2FyLTU2Ny5wZGZVVA0AB2paMmdsWjJnaloyZ3V4CwABBPUAAAEEFAAAAFBLBQYAAA
AAgACAMoAAADYAAAAAA= </fileBinary>
        <fileExtension>ZIP</fileExtension>
    </attachment>
    <additionalLines>
        <additionalLine>
            <lineNumber>2</lineNumber>
            <lineAdditionalData>
                <dataName>N00001_GARANCIA</dataName>
                <dataDescription>Mákdáraló garanciajegy</dataDescription>
                <dataValue>N00001|Fmakdaralo-garjegy-123.pdf|T12 hó</dataValue>
            </lineAdditionalData>
        </additionalLine>
    </additionalLines>
</receiptAdditional>
```



```
</lineAdditionalData>
</additionalLine>
<additionalLine>
    <lineNumber>6</lineNumber>
    <lineAdditionalData>
        <dataName>N00001_GARANCIA</dataName>
        <dataDescription>Bifilátor garanciajegy</dataDescription>
        <dataValue>N00001|Fbifliator-gar-567.pdf|T24 hó</dataValue>
    </lineAdditionalData>
</additionalLine>
<additionalLine>
    <lineNumber>9</lineNumber>
    <lineAdditionalData>
        <dataName>N00001_GARANCIA</dataName>
        <dataDescription>Borotva garanciajegy</dataDescription>

<dataValue>N00001|Uhttp://filipsz.hu/warranty/getPdf?id=abcd1234defg6789 |T18
hó|INe felejtse félévente ellenőriztetni a kések élességét.</dataValue>
    </lineAdditionalData>
</additionalLine>
</additionalLines>
</receiptAdditional>
```

13.8.2 Coupon information

The name of the coupon information field (dataName field): N00002_KUPON

The content of the dataValue field may include the following fields as described in the table below:

Field id	Data description
C	The unique text code of the coupon.
S	The name of the company, store, retail chain, etc., issuing the coupon. It does not have to match the header data of the receipt.
D	A short textual description of the coupon.
E	The expiration date of the coupon, in the format “YYYY-MM-DD”.
T	The validity period of the coupon with a unit of measurement, e.g., “1 month”, “2 weeks”.
I	Other textual information.

Interpretation example:

A half-price Magnum ice cream coupon issued by Nudli retail chain, valid for one week. It is associated with the entire e-receipt, not with a specific item. The coupon cannot be combined with other discounts.

```
<receiptAdditional>
    <documentNumber>NY-T00100001/12345678/0011/00022</documentNumber>
    <additionalHead>
        <AdditionalData>
            <dataName>N00002_KUPON</dataName>
            <dataDescription>Magnum 50% kedvezménykupon</dataDescription>
            <dataValue>N00002|CMGNM500FF|SNudli|DMagnum kupon|T1 hétköznap|I A kupon más
kedvezménnyel nem vonható össze.</dataValue>
        </AdditionalData>
    </additionalHead>
```



```
...  
</receiptAdditional>
```

13.8.3 Quick access barcode

In certain cases, it may be necessary to display data linked to a receipt in an easily accessible way in the customer application. An example of this is the barcode that opens the exit gate in self-checkout counters, which the customer must be able to view in the customer application as quickly as possible after the purchase. The application may even display such marked codes on the notification “badge” shown on the locked screen of the customer’s device, so that unlocking the device is not required for exit.

The quick access barcode (dataName field): N00003_GYORSKOD

The dataValue field describes its display format (e.g. Code 128, data matrix, QR code) and their required and optional settings according to the table below:

Data identifier	Data description
T	Possible types and values of the barcodes (mandatory field): <ul style="list-style-type: none">„1D” – 1D barcode, subcategory type to be provided with „S” identifier.„DM” – Data Matrix„P4” – Pdf417 2D barcode„QR” – QR-code
S	The subtype of the barcode, interpreted only in the case of 1D barcodes, possible values: UPC_A, UPC_E, EAN13, EAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1_128, GS1_DB_O, GS1_DB_T, GS1_DB_L, GS1_DB_E
C	Number of columns, interpreted only in the case of Data Matrix (“DM” type) or Pdf417 (“P4” type)
R	Number of rows, interpreted only in the case of Data Matrix (“DM” type) or Pdf417 (“P4” type)
E	Error correction mode: <ul style="list-style-type: none">In the case of QR code (“QR” type), its value may be L, M, Q, or HIn the case of Pdf417 (“P4” type), it can be a number between 0 and 8
M	Encoding mode, can only be specified in the case of Pdf417 (“P4” type), its value may be 0 (binary), 1 (ASCII text), or 2 (numeric). In the case of QR code, only binary mode is supported, so it does not need to be specified separately.
D	The data to be displayed in the code (mandatory field), with content consistent with the supplementary parameters (e.g. encoding, content limitations, dimensions).

Interpretation example: Code 128 type (exit code)

```
<receiptAdditional>  
  <documentNumber>NY-T00100001/12345678/0011/00022</documentNumber>  
  <additionalHead>  
    <AdditionalData>  
      <dataName>N00003_GYORSKOD</dataName>  
      <dataDescription>Kapukód</dataDescription>  
      <dataValue>N00003|T1D|SCODE128|D1234567890123456</dataValue>  
    </AdditionalData>  
</receiptAdditional>
```



```
</additionalHead>  
...  
</receiptAdditional>
```

13.8.4 Line – or dot code

All line or dot codes to be displayed as a supplement to the receipt must have the prefix “N00004_” written into the dataName field. After the prefix, any text meeting the validation condition may be written.

The dataValue field describes the display method, which is the same as described in the “Quick access barcode” section.

Interpretation example

```
<receiptAdditional>  
    <documentNumber>NY-T00100001/12345678/0011/00033</documentNumber>  
    <additionalHead>  
        <AdditionalData>  
            <dataName>N00004_UTMUTATO</dataName>  
            <dataDescription>Cipőápolási útmutató</dataDescription>  
            <dataValue>N00004|TQR|EQ|Dhttps://barefoot-  
cipok.hu/apolas.pdf</dataValue>  
        </AdditionalData>  
    </additionalHead>  
...  
</receiptAdditional>
```

13.8.5 Image

The receipt or its items can be illustrated with small images, which can be attached as an attachment to the receipt. The attachment may be the image file itself, or several files in a common zip archive. The zip file may contain both a warranty card and an image file, but they must be referenced in separate supplementary information elements so that their display in the customer applications occurs in the correct place and manner.

The images attached to the receipt can be displayed by the customer application among the supplementary data of the receipt or the item.

All images to be displayed as a supplement to the receipt must have the prefix “N00005_” written into the dataName field. After the prefix, any text meeting the validation condition may be written.

The content of dataValue may include the fields described in the table below:

Data identifier	Data description
F	The filename of the image attached as a file
I	Other text information



Attaching an image promoting the store's promotion to the receipt:

```
<receiptAdditional>
    <documentNumber>NY-T00100001/12345678/0011/00033</documentNumber>
    <additionalHead>
        <AdditionalData>
            <dataName>N00005_PROMO</dataName>
            <dataDescription>Nyári akcióink</dataDescription>
            <dataValue>N00005|Fnyari-promo.png</dataValue>
        </AdditionalData>
    </additionalHead>
...
</receiptAdditional>
```

13.8.6 General file attachment reference

The attachment added to the document does not need to serve a specific function, like a warranty card or image illustration; it may contain general supplementary information about the seller, a promotion, the transaction recorded on the receipt, or some of its items.

Customer applications can fully and correctly display attachments if the attached file or files are referenced in the supplementary data (additionalData) submitted for the receipt or its items. Without a reference, the customer application can only display a generated filename based on the extension, or in the case of a ZIP package, it can display the files within it among the other receipt data. Attachments can only be linked to receipt items using the additionalData supplementary element.

The use of general file attachment reference allows customer applications to display the attached files alongside the document or its items, accompanied by a short text explanation and the exact filename.

A file — which can be the only attachment of the document, or any file packaged in the attached ZIP — can be referenced from several supplementary elements (additionalData) within one receipt, for example, a brochure for skincare advice attached may be referenced by the supplementary information elements of two separately listed body lotion items.

The name of the attachment reference (dataName field): N00006_MELLEKLET

The content of dataValue may include the field described in the table below:

Data identifier	Data description
F	The filename of the attached attachment (mandatory).

The content description of the attachment goes into the dataDescription field, it is advisable to provide short and concise text so that customer applications can display it in full on the screen of the mobile device.

Interpretation example 1

We attach a shoe care guide in PDF format as an attachment to the product recorded as item 3.

Customer envelope (CustomerDocumentType) supplementary content:

```
<receiptAdditional>
    <documentNumber>NY-T00100001/12345678/0023/00045</documentNumber>
    <attachment>
        <fileBinary>SWR1IGt1cs08bCBhIGbDoWpsIGJpbsOhcm1zYQ==</fileBinary>
```



```
<fileExtension>PDF</fileExtension>
</attachment>
<additionalLines>
    <additionalLine>
        <lineNumber>3</lineNumber>
        <lineAdditionalData>
            <dataName>N00006_MELLEKLET</dataName>
            <dataDescription>Így ápol a cipődet</dataDescription>
            <dataValue>N00006|Fcipőápolás.pdf</dataValue>
        </lineAdditionalData>
    </additionalLine>
</additionalLines>
</receiptAdditional>
```

Interpretation example 2

A PDF attachment for each of receipt items 3 and 5, attached in a zip package.

The two PDFs – “ABC_kezelesi_utmutato.pdf” and “CDE_osszetevok.pdf” – are placed into a common compressed file, in the root of the zip (without subdirectories).

In the customer envelope (CustomerDocumentType)

```
<receiptAdditional>
    <documentNumber>NY-T00100001/12345678/0067/00089</documentNumber>
    <attachment>
        <fileBinary>
UEsDBBQACAAIAFGja1kAAAAAAAAAAAAZACAAQUJDX2t1emVsZXNpX3V0bXV0YXRvLnBkZ1
VUDQAHaloyZ4RNpGjzTaRodXgLAAEE9QEAAAQAAAAbcwhwceMCAFBLBwh3hL1kBwAAAAUAAABQ
SwMEFAAIAAgAR6NrWQAAAAAAAABQAAABIAIBDREVfb3NzemV0ZXZvay5wZGVVA0AB1ZaMm
eFTaRoJU6kaHV4CwABBPUAAAEEFAAAAMHMIcHHjAgBQSwcId4S5ZAcAAAAFAAAAUEsBAhQDFAAI
AAgAUaNrWXeEuWQHAAAABQAAABkAIAAAAAAAKSBAAAAEEFCQ19rZXp1bGVzaV91dG11dG
F0by5wZGVVA0AB2paMmeETaRo802kaHV4CwABBPUAAAEEFBLAQIUAxQACAAIAEejal13
hL1kBwAAAAUAAAASACAAAAAAACkgW4AAABDREVfb3NzemV0ZXZvay5wZGVVA0AB1ZaMm
eFTaRoJU6kaHV4CwABBPUAAAEEFBLBQYAAAAAAGACAMcAAADVAAAAAA=
        </fileBinary>
        <fileExtension>ZIP</fileExtension>
    </attachment>
    <additionalLines>
        <additionalLine>
            <lineNumber>3</lineNumber>
            <lineAdditionalData>
                <dataName>N00006_MELLEKLET</dataName>
                <dataDescription>Használati utastás</dataDescription>
                <dataValue>N00006|FABC_kezelesi_utmutato.pdf</dataValue>
            </lineAdditionalData>
        </additionalLine>
        <additionalLine>
            <lineNumber>5</lineNumber>
            <lineAdditionalData>
                <dataName>N00006_MELLEKLET</dataName>
                <dataDescription>Termékösszetevők</dataDescription>
                <dataValue>N00006|FCDE_osszetevok.pdf</dataValue>
            </lineAdditionalData>
        </additionalLine>
    </additionalLines>
</receiptAdditional>
```



14 Data transfers between customer app and E-Cash Register

In the e-receipt system, asymmetric keys used for encrypting receipts can be generated either in the customer application or in the e-cash register. In both cases, the key must be transferred to the other party involved in the transaction.

The key generation and transfer occur at different stages of receipt creation, depending on the case:

1. If the key is generated in the customer application, it can be transferred within a specified time frame, starting from the creation of the receipt up to just before its closure. The key transfer may open a new sales receipt, followed by the addition of items, payment, and receipt closure. Alternatively, it is permitted for the e-cash register to accept the encryption key from the customer application after adding items and processing payment but before finalizing the receipt.
2. If the customer does not provide an encryption key, the e-cash register must generate one at the moment of finalizing the receipt, provided that a specific condition is met. The key is then transferred to the customer either through printing on a receipt copy or displaying it on the customer screen.

The e-cash register must be designed to ensure the customer can always access the receipt. The second case requires a simpler logic—when closing the receipt, the system checks whether the customer has provided an encryption key. If not, the e-cash register generates the key pair, forms the retrieval key, etc.

In the first case, however, certain situations may arise where improper handling could prevent the customer from accessing the receipt data, such as:

- The customer presents a new QR code during receipt creation because they forgot to include a coupon in the previous one. The new QR code contains a new encryption key. The e-cash register must determine which key to use for encryption.
- The customer initiates a transaction by presenting a QR code, causing the e-cash register to open the receipt, but then decides to step out of line to make additional purchases. The next customer does not present a key, and the e-cash register still remembers the previously presented key. As a result, the e-receipt is encrypted with the wrong key, and the actual buyer cannot decrypt the receipt data because their key was not printed on the receipt copy.

Resolving such situations may require the intervention of the e-cash register operator. The design of e-cash registers should enable them to prevent the creation of encrypted e-receipts that cannot be unlocked, either automatically or with cashier assistance.

The use of NFC allows for bidirectional data exchange between the customer's mobile device and the e-cash register. This technology enables the customer to verify the e-cash register's authenticated data ("identity"), allows the e-cash register to confirm the received encryption key and other data, and even facilitates the transfer of the entire receipt envelope.

This section outlines the expected handling of key and data transfers under normal and exceptional circumstances.



14.1 Data transfer from the customer application for an open receipt

As described in the introduction of this section, a specific time frame is allocated for transferring the encryption key and other customer-related information—such as invoice request, billing address, and gratuity.

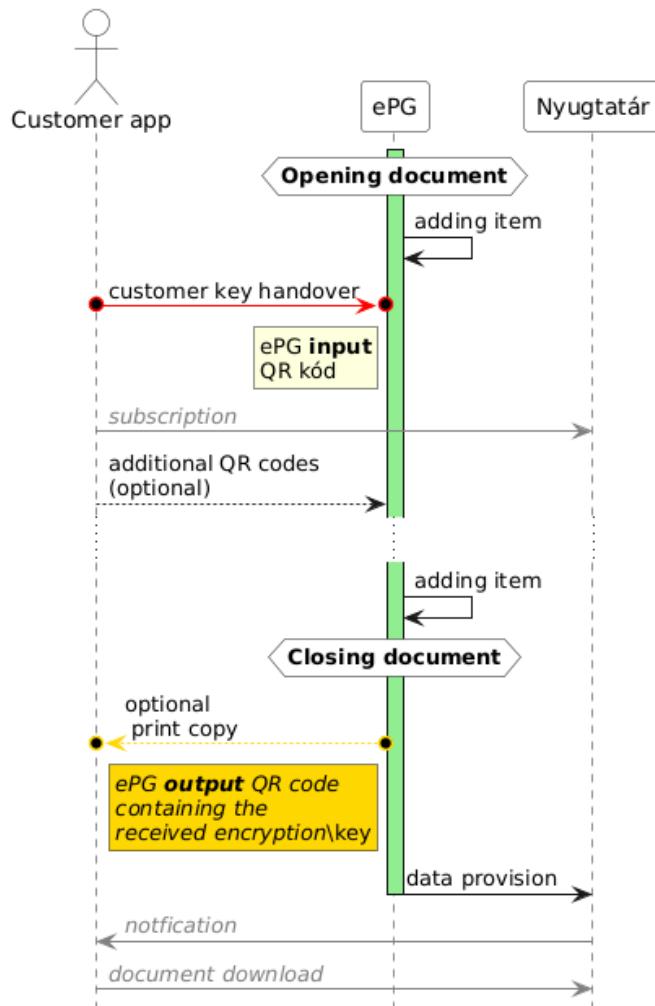
Data Transfer Rules:

- The earliest possible transfer occurs at the beginning of the sales transaction, where the encryption key provided by the customer opens the receipt.
 - If the e-cash register is configured to accept customer data even without an open receipt, then the receipt must be opened immediately.
- During and after adding receipt items, as well as during and after recording payment methods, the system must allow the transfer of customer data.
 - The customer may present multiple QR codes (not just segmented ones). The e-cash register must accept and process all of them.
 - Not every data package (QR code) from the customer application contains an encryption key.
 - If the customer has already provided a QR code with an encryption key and later presents another QR code with a new key, the e-cash register must accept the latest key and use it for encrypting the receipt.
 - Data transfer from the customer application may also occur without providing any encryption key.
- If a receipt is interrupted after receiving customer data, a zero-total e-receipt must be submitted to the e-receipt archive, containing both the recorded items and their invalidation, just like a normal e-receipt. The printed receipt copy (if requested) must include the text "Interrupted Receipt.".
- This rule also applies when the receipt is opened immediately after receiving customer data but must be canceled before adding any items. In this case, a zero-total e-receipt must be submitted with the "Interrupted Receipt" text.
 - If this feature is supported by the e-cash register distributor, the cancellation function must be easily accessible to the cashier.
- The customer application should minimize both the reuse of encryption keys and the generation of unnecessary encryption keys that are never read by the e-cash register.
 - When using QR codes, their display time should be limited—each encryption key can be displayed on the customer's mobile device screen only once, for a maximum of 15 seconds.
 - The customer application may attempt to retrieve the receipt using the search key associated with each generated and displayed encryption key, as described in the "Receipt Retrieval" subsection.
 - The e-cash register will display the encryption (public) key received from the customer application in the output QR code. The customer application can optionally scan this from the display or printed receipt copy, allowing the customer to verify which encryption key the e-cash register used—even if multiple keys were provided for the transaction.

14.1.1 QR Code data transfer process

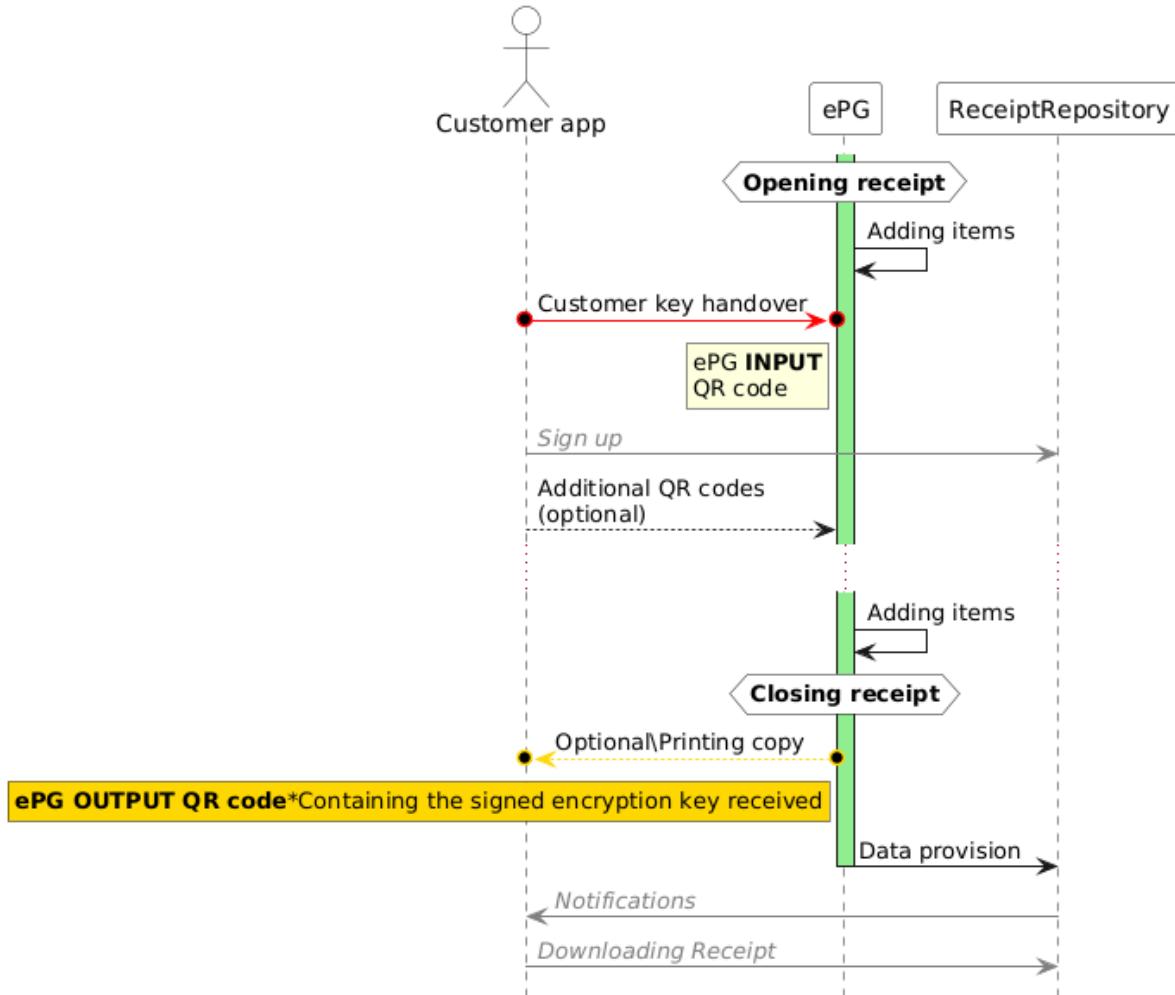
After opening the receipt and recording items, the e-cash register scans the QR code containing the encryption key from the customer application. Upon closing the receipt, the e-cash register displays the output QR code on the customer display or prints it on the receipt copy. This output QR code contains the encryption (public) key extracted from the customer application, digitally signed.

Key handover with QR code from Customer application
[For opened document]
Confirmed upon document closure



The receipt is opened when the QR code is scanned from the customer application, and only after that are the receipt items added.

**Key transfer with QR code from Customer Application
[Open Receipt]\Confirmed when closing receipt**



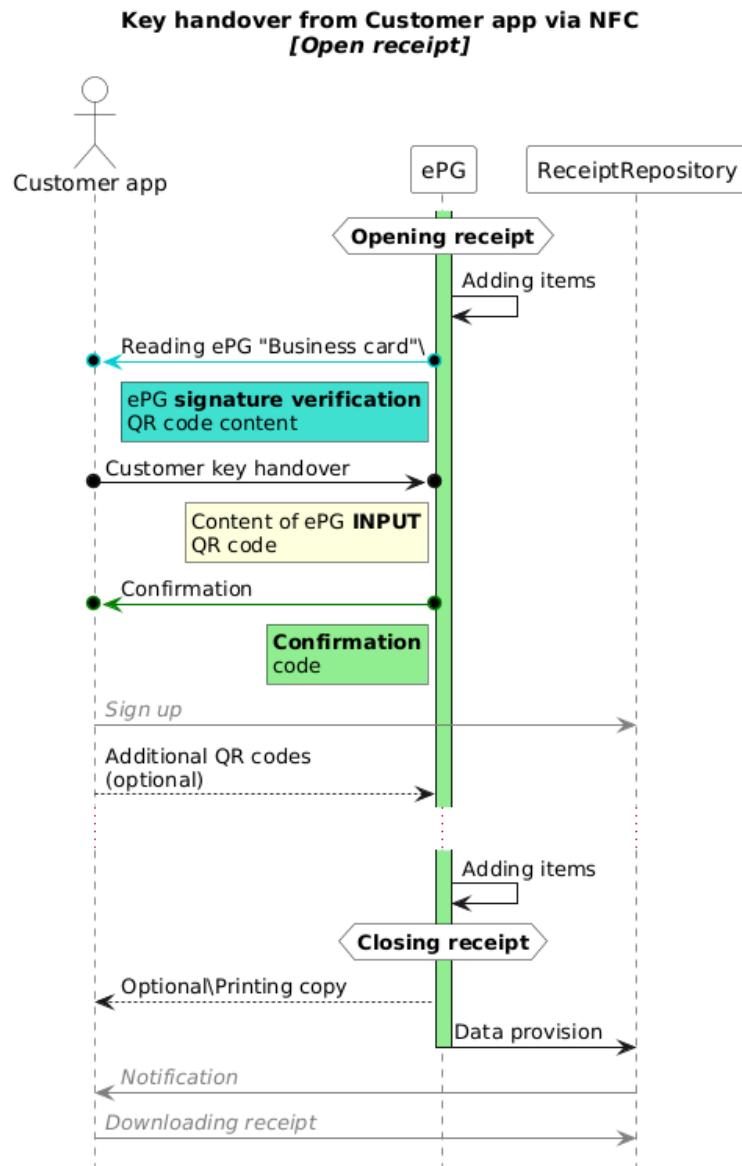
14.1.2 Data transfer via NFC

The NFC message exchange occurs after the receipt is opened and the items are recorded in the customer application, allowing the encryption customer key to be read in NDEF record format by the e-cash register.

Steps of NFC message exchange:

- The customer application reads the "business card" of the e-cash register, which matches the so-called "signature verification QR code." This contains the basic details of the e-cash register, digitally signed with the e-cash register's signing key. Using the extracted data, the customer application can verify that it is exchanging messages with a legitimate e-cash register using a valid certificate.
- Then, the e-cash register transmits an NDEF record containing the same data set as the "e-cash register input QR code." This includes all necessary information, such as the encryption key, invoice request, billing address, coupon codes, etc.
- The e-cash register confirms the received data in a digitally signed NDEF record.

This three-step data exchange transaction can be completed in a single approach without removing the device.



14.2 Data transmission at receipt closure

At the moment of receipt closure, the e-cash register must check whether it has received an encryption key from the customer application. If it has received one, it must use that key in the receipt envelope to encrypt the symmetric key used for receipt data encryption. If no encryption key was received, the e-cash register generates its own key pair and must provide the private key to the customer application.

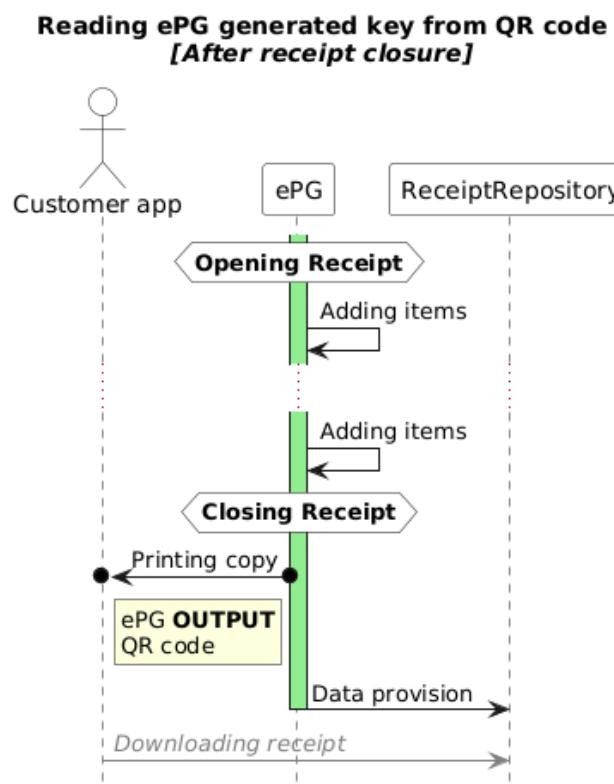
Rules for data transmission:

- At the moment of receipt closure, the e-cash register generates an "output QR code," which either contains the received public encryption key or the self-generated private key. In the first case, it also serves as confirmation that the received key was used.

- The output QR code must be displayed on the customer screen for a predetermined period or until an event occurs, such as:
 - A set duration, up to a maximum of one minute,
 - The opening of the next receipt or
 - Manual deletion by the cashier.
- The output QR code must be printed on an optionally requested receipt copy for the customer.
- Before opening a new receipt, at the customer's request, the last receipt's output QR code content must be retrievable.
- During the display period of the output QR code (including any customer-requested re-display), if the e-cash register has the necessary peripheral, NFC readability must also be provided.
- The customer application may request the full receipt envelope from the e-cash register using the NFC-extracted output code. If the receipt fits within the connected peripheral's capacity, it must be transmitted in NDEF record format.

14.2.1 Transmission via QR code or NFC

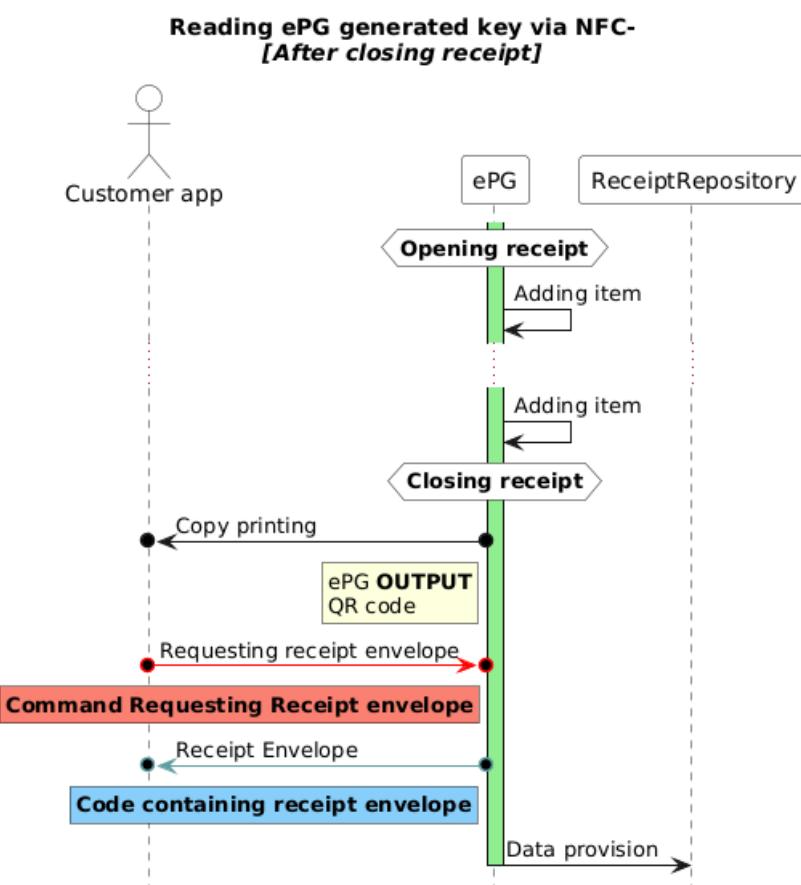
At the time of receipt closure, the e-cash register displays the output QR code on the customer screen and, if an NFC-compatible peripheral is connected, makes the same content available via NFC. The customer application reads the data but does not initiate further data transmission.



14.2.2 Retrieving the receipt envelope via NFC

This process differs from the one described in the previous section in that, after reading the output code via NFC, the full receipt envelope is requested in the following steps:

- Reading the NDEF record containing the output code.
- Writing an NDEF record requesting the transfer of the receipt envelope.
- The cash register receives the request and composes a response NDEF record, which contains either:
 - The receipt envelope, or
 - An error response if there is a technical obstacle, such as the size being too large to fit within the NDEF record supported by the connected NFC peripheral.
- The customer application reads the response NDEF record made available by the cash register.
- In the case of successful reading, the customer application will no longer call the e-Receipt system's document retrieval endpoint.





15 Other restrictions on the operation of E-Cash registers

15.1 Printing

The printer of the e-cash register is used to generate paper-based receipt copies. Printing capability is a mandatory function of e-cash registers, but in the event of a printer failure, the e-cash register can continue to operate. In such cases, it is the operator's responsibility to restore the printing capability as soon as possible.

15.2 Relationship between fiscal day and calendar day

Annex 2, Part B, Section 5 of the Regulation specifies the following technical requirement for all e-cash registers (both hardware-based and cloud-based):

"A fiscal day cannot end on a later calendar day than the one on which it began."

The purpose of this provision is to ensure that the content of receipts and daily sales reports issued by the e-cash register aligns with the actual date of issuance.

The Regulation does not explicitly define a required operational method to enforce this rule. The distributor may determine the technical solution based on the design and expected usage conditions of the e-cash register type they intend to approve.

If it is clearly impossible for any unit of the approved e-cash register type to be used around midnight, no special operational feature needs to be implemented to enforce this rule. The chosen solution must be specified in the approval application (Annex 1, Section 1.10.5 of the Regulation).

For e-cash register types where receipts may be issued around midnight, the following technical solutions can ensure that the fiscal day ends on the same calendar day it began:

- a) The e-cash register prevents new receipt issuance after a specified time before midnight.
- b) The e-cash register automatically terminates any ongoing receipt issuance before midnight and closes the fiscal day automatically

The time settings for these operations should be determined based on the usual duration of receipt issuance and the time required for termination and closure, aiming to minimize the operational downtime around midnight.

It does not violate the Regulation if the e-cash register automatically opens a new fiscal day after midnight and records any interrupted transaction as a new receipt. However, implementing this feature is optional.

Multiple fiscal days within a single calendar day are not prohibited if justified by the operational needs of the business.

15.3 In-store cash withdrawal (cash back)

In the e-receipt system, cash-back transactions can be recorded as follows:

- During receipt issuance, the e-cash register verifies the eligibility for cash withdrawal (e.g., minimum transaction amount, cash withdrawal limit) via a connected PWCB (Pay With Cash Back) enabled bank card terminal.
- The receipt should only reflect the amount related to the purchase, excluding the cash withdrawal portion of the transaction.



- The total card charge and the receipt total difference must be recorded on a separate cash movement receipt, as follows:
 - Transaction title: "Cash Withdrawal" (Code 42)
 - Receipt must include a bank card payment entry equal to the cash withdrawal amount and a corresponding cash withdrawal entry

Example Transaction:

The customer purchases goods for 4,200 HUF
They request a 10,000 HUF cash withdrawal via a card transaction.

Receipt Details, amounts on the receipt:

- Total amount: 4 200 HUF
- Payment Method (Register Balance Change):
 - Bank card: 4 200 HUF
- No change is displayed.

Cash Movement Receipt Details:

- Transaction Title: "Cash Withdrawal" (XML Code 42)
- Deposits:
 - Bank card: 5 800 HUF
- Withdrawals:
 - Cash: 5 800 HUF

15.4 Displaying VAT group information

If the e-cash register operator (or, in the case of dual-operator e-cash registers, the fuel owner or gas station operator) belongs to a VAT group, the electronic receipts and their printed copies must display both the individual tax number (xxxxxxxx-4-x) and the VAT group identifier number (xxxxxxxx-5-xx).



16 Master data

The master data that e-cash registers can query from the NAV can be found in the "[Product Master Data Query](#)" section.

This section contains master data that is not queryable via the interface but is publicly available.

16.1 Competence codes indicating the relevant State Tax Authority (countyCode)

	Corporate business county code	Sole proprietorship county code
Baranya county	02	22
Bács-Kiskun county	03	23
Békés county	04	24
Borsod-Abaúj-Zemplén county	05	25
Csongrád -Csanád county	06	26
Fejér county	07	27
Győr-Moson-Sopron county	08	28
Hajdú-Bihar county	09	29
Heves county	10	30
Komárom-Esztergom county	11	31
Nógrád county	12	32
Pest county	13	33
Somogy county	14	34
Szabolcs-Szatmár-Bereg county	15	35
Jász-Nagykun-Szolnok county	16	36
Tolna county	17	37
Vas county	18	38
Veszprém county	19	39
Zala county	20	40
North-Budapest	41	41
East-Budapest	42	42
South-Budapest	43	43
Pest County and Capital City Directorate of Key Taxpayers	44	44
Taxpayers with Exclusive Jurisdiction	51	51



16.2 Country code type according to ISO 3166 alpha-2 standard

The official international ISO standard country code list can be found at the following link, where the Alpha-2 code column should be considered. <https://www.iso.org/obp/ui/#search>

16.3 Postal code database availability

<https://www.posta.hu/szolgaltatasok/iranyitoszam-kereso>

https://www.posta.hu/static/internet/download/Iranyitoszam-Internet_uj.xlsx

16.4 VTSZ (Customs Tariff Number) database availability

https://nav.gov.hu/pfile/file?path=/szabalyzok/tajekoztatasok/4002_2019._1.melleklet

16.5 SZJ (Service Classification Number) database availability

https://www.ksh.hu/osztalyozasok_teszor2-1

16.6 KN (Combined Nomenclature) database availability

https://www.ksh.hu/kombinált_nomenklatura

16.7 CSK (Packaging Material Catalogue Code) database availability

http://njt.hu/cgi_bin/njt_doc.cgi?docid=142904.348985 (Annex 1, section A)

16.8 KT (Environmental Product Code) database availability

http://njt.hu/cgi_bin/njt_doc.cgi?docid=142904.348985 (Annex 1, section B)

16.9 EJ (Building Register Number) database availability

https://www.ksh.hu/epitmenyjegyzek_menu

16.10 TESZOR (Product and Service Classification System) database availability

https://www.ksh.hu/osztalyozasok_teszor2-1



17 Accessibility of the environments

The system operates with two environments. Production e-cash registers and customer applications can only access the live production environment.

For AE manufacturers, e-cash register distributors, and customer application developers, a testing (inspection) environment is available. This environment allows for the testing of cash registers and customer applications under development. Additionally, this environment is used for the approval process of e-cash registers and customer applications.

17.1 Inspection environment

17.1.1 URLs of the inspection environment

Interface	URL	Accessible from
NAV-I unauthenticated	https://navi-bv.enyugta.nav.gov.hu	Closed APN for hardware-based AEs
NAV-I authenticated	https://navi-bv-sec.enyugta.nav.gov.hu	Closed APN for hardware-based AEs
FAM unauthenticated	https://fam-bv.enyugta.nav.gov.hu	Internet
FAM authenticated	https://fam-bv-sec.enyugta.nav.gov.hu	Internet
Nyugtatár (Receipt Repository)	https://lekerdezo-adm-bv.enyugta.nav.gov.hu	Customer app backend fixed IP address
Nyugtatár (Receipt Repository)	https://lekerdezo-bv.enyugta.nav.gov.hu	Internet

17.1.2 Development support resources

GitHub

The NAV publishes the following resources on its official GitHub page (<https://github.com/nav-gov-hu/eRECEIPT>):

- XSD
- XSD HTML-based visualization
- This Developer Documentation

KOBAK Portal

The development and testing processes are supported by the web-based environment, the ePG Portal, which can be accessed at:

- <https://kobakonline.nav.gov.hu/>

Apidog collection

The sample calls for the endpoints of the "[Business Services Provided by NAV for E-Cash registers](#)," "[Services provided by the Receipt Repository](#)" and "[Cloud-based Fiscal Module \(FAM\)](#)" sections are published by NAV on its official GitHub page in the form of an Apidog collection.



17.2 Production environment

17.2.1 URLs of the production environment

Interface	URL	Accessible from
NAV-I unauthenticated	https://navi.enyugta.nav.gov.hu	Closed APN for hardware-based AEs
NAV-I authenticated	https://navi-sec.enyugta.nav.gov.hu	Closed APN for hardware-based AEs
FAM unauthenticated	https://fam.enyugta.nav.gov.hu	Internet
FAM authenticated	https://fam-sec.enyugta.nav.gov.hu	Internet
Nyugtatár (Receipt Repository)	https://lekerdezo-adm.enyugta.nav.gov.hu	Customer app backend fixed IP address
Nyugtatár (Receipt Repository)	https://lekerdezo.enyugta.nav.gov.hu	Internet

18 Helpdesk and technical support

This chapter provides guidance for troubleshooting and obtaining further assistance.

18.1 Helpdesk availability

Two separate helpdesks are available for resolving issues and answering questions related to the system. For all inquiries and problems related to the live system, users can submit a request through <https://www.nav.gov.hu/nav/e-ugyfsz/levelkuldes> under the subject "e-Cash Register - IT Issues."

For technical support specifically related to the test system and only concerning interface services for developers, assistance is available through the designated issue tracking page on GitHub at <https://github.com/nav-gov-hu/eReceipt/issues> or by sending an email to init.epg.helpdesk@nav.gov.hu.

19 Version tracking

To facilitate easier tracking of service modifications, this chapter includes major changes and the different introduced interface versions.

When services, especially interfaces, undergo modifications, a new Developer Documentation will be issued.

If a change is backward-compatible with the previous version, the main version number will remain unchanged, and there will be no need to modify the version number in the context root.

From the perspective of the service, the version is defined by the URL, while from the perspective of the business data model, the version is defined by the value of the requestVersion tag specified in the HTTP body (in XML).

Major versions are those in which backward compatibility of the business data model between versions cannot be ensured. Minor versions are those in which, within a given major version, the business data remains compatible.



Each new major version is always associated with new URLs and a new XML namespace. Minor versions inherit the URL and namespace data of the major version to which they belong.

Considering that during the transition period defined in the legislation two different versions of the XML API operate in parallel; it is necessary to stipulate some general correlations regarding versions.

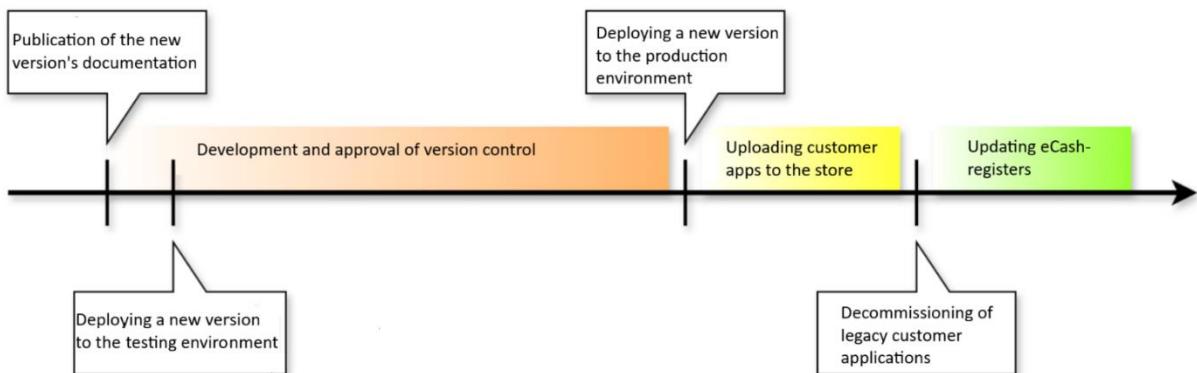
These correlations are as follows:

1. There is no crossover between versions. The version in the Context root must always match the XML namespace version. (For example, XML with version v2 cannot be submitted to a URL containing a v1 context root.)
2. Every successfully received receipt can only be downloaded if the requestVersion value of the query matches the value of the currently supported major version. However, there is no crossover upward between major versions: clients using a lower version can only download receipts with requests matching their own major version. (For example, 3.x receipt downloads have full visibility of 1.x, 2.x, and 3.x receipts, while 2.x queries cannot see 3.x receipts.)
3. Every receipt submitted with a /document or /report request can only be downloaded with at least a query using the requestVersion corresponding to the submission, never with a lower one. (For example, a receipt submitted on the 2.0 /document endpoint can be downloaded with a 3.0 query, but not the other way around.)
4. For every base receipt (cash receipt, simplified invoice, or invoice), only such a modifying or canceling receipt may be submitted whose requestVersion value is at least equal to the envelopeVersion value of the base receipt, never lower. (For example, a 3.0 canceling receipt can be submitted for a 2.0 base receipt, but not the other way around.)

In accordance with the above, version changes require a coordinated process for the entire system, the main steps of which in chronological order are as follows:

1. The new version and the corresponding Developer Documentation are published on the NAV GitHub page.
2. After this, the central components handling the new version are installed in the test environment.
3. Vendors have the time defined in the Regulation to adapt to the changes and complete the testing and authorization process of the new version. It is especially important for the distributors of customer applications to complete development and authorization within the deadline, since customer applications using the old interface version cannot download receipts submitted by e-cash registers on the new version of the interface.
4. NAV installs the components using the new version in the live environment.
5. After this, the updated and authorized customer applications can be uploaded to the stores.
6. NAV withdraws the old versions of customer applications. If the distributor of the customer application does not carry out the update within the available time, its users will not be able to download receipts after the version change.

7. After the customer applications are updated, the new software version can be deployed to the e-cash registers. If the modification also affects the FAM version, FePG applications can only be updated from this point on as well.



19.1 Version 1.0

At the time of the document's publication, the value "1.0" must be included in the header/requestVersion element.

19.2 Version 1.1

From September 19, 2025, the use of the following versions is mandatory.

e-Cash register interface

requestVersion	1.1
Context root version number	v1
envelopeVersion	1.1

Nyugtatór interface

requestVersion	1.1
Context root version number	v1
envelopeVersion	1.1

FAM interface

requestVersion	1.0
Context root version number	v1
envelopeVersion	1.0



Nemzeti Adó-
és Vámhivatal
