Nemzeti Adó- és Vámhivatal

NAV API Gateway
M2M REST API, Authentication Specification and Developer's Documentation

Nemzeti Adó- és Vámhivatal

Table of Contents

Nemzeti Adó- és Vámhivatal

Glossary & Abbreviations

Term	Description
Taxpayer	The registered taxpayer in Hungary on whose behalf the users act.
Signature Key	For the purposes of the present document, a signature key shall be defined as a string of characters used to amend – "sign" – another string of characters or symbols.
API	Application Programming Interface.
BASE64	A content-encoding system based on a 64-character alphabet, used to turn data containing binary and special characters into ASCII character strings (Binary-to-text encoding, RFC 3548).
M2M	Machine-machine connection.
NAV	The National Tax and Customs Authority.
Online Invoice System	The system performing the processing of online invoice data provisions. https://onlineszamla.nav.gov.hu/
Manufacturer	A natural or legal person developing the program that implements API communication.
SHA-512	A 512-bit secure HASH algorithm (Secure Hash Algorithm 3, RFC 6234).
SHA3-512	A 512-bit secure HASH algorithm (FIPS 202)
Technical User	A user required for communication via the REST API, which can be created by the Primary User in the Online Invoice System.
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/).
REST	A Representational State Transfer (REST) web service, also called RESTful.
Primary User	The user of the system who is the Taxpayer themselves or their statutory or permanent representative, and, in this capacity, bears full authority to use the system. (The sole exception to this authority is that of data traffic via the REST API, which can only be performed via a technical user created by the Primary User.)
Endpoint	A path to access the service provided by the operation.
Single Sign-On (SSO)	A web-based single sign-on method, which is a special form of software authentication that allows the user to authenticate only once when logging into a given system, and afterwards, they can access all system resources and services without further authentication.
Operation	A collective term of information technology procedures and services that can be called via the offered REST web service.
Web Service	A collection of protocols and standards for inter-application data exchange.

Document History

Date	Author	Version	Change
04/05/2023	N.Á.	0.1	Initial version
06/06/2023	N.Á.	0.2	Version intended for first issue
14/07/2023	N.Á.	0.3	Minor fixes

1 INTRODUCTION

The types and elements that implement generic API communication have been previously extracted into a common XSD (common.xsd). This allows the communication of APIs used by multiple NAV projects to be unified using this schema definition.

For this purpose, a central API Gateway application has been created, which acts as a frontend, performing the authentication and authorisation of requests and request distribution.

In the first phase of development, the Gateway will exclusively serve the eVAT M2M system. This document describes the generic authentication method to be used on the eVAT machine interface. The authentication and authorization mechanisms used on the machine interface are based on the types defined transparently for developers in the common schema, and use the same authentication error codes as those already established in the Online Invoice System.

1.1 Purpose

The purpose of the present document is to describe how the authentication of communication through an M2M connection works, to present the XML message structures used by it, and to provide support for integration with the interfaces of taxpayers' own systems.

This document includes the business and technical content for the following schema definitions:

Schema	Content
common.xsd	Generic types, catalogue elements and primitives describing NAV communication

1.2 Conditions of use for taxpayers

In order to implement M2M communication, the taxpayer must have a technical user with the appropriate permissions.

The creation and setup of a technical user can be performed following the steps below:

1) The taxpayer, legal representative or permanent representative must log into the Online Invoice System at least once.

2) A taxpayer wishing to use the service must first create a technical user or set the appropriate permissions for an existing technical user on the user editor interface of the Online Invoice System, i.e. the technical user created for communication with the Online Invoice System interface can also be used. M2M communication is only possible via a technical user. The taxpayer can choose the number of technical users to create for M2M communication. Only primary users can create technical users and set their permissions.

3) A signature key and exchange key must be generated for the technical user on the interface developed. The signature key is required for calculating the requestSignature value, which is needed for signing messages; the exchange key is required for the server-side encoding and client-side decoding of the possible data transfer token.

Assignment of permissions:

In the case of eVAT, a multi-level permission system has been developed, where is interface access permission is on the first level. The second level contains the specific permissions of the eVAT interface, which also need to be set by primary users according to the operations they want to use. Without the appropriate permissions, all requests from the client will be rejected with the corresponding error code, which can be found in the section "**ERROR PROCESSING"**.

Permissions can be set for each technical user as follows:

1) The primary users must assign the necessary permissions to technical users on the web interface developed for this purpose. When logged into the web interface of Online Invoice, the permissions of the given technical user can be controlled in the Users menu, including whether it can access the machine interface of eVAT.

2) In addition, the permissions applicable within the eVAT machine interface need to be set by switching to the eVAT web interface. On web interfaces, the single sign-on (SSO) method allows users to authenticate only once.

The above-mentioned requirements apply per system level. Technical users and keys created and generated in the test environment cannot be used in the production environment, similarly to the Online Invoice System.

1.3 Technologies to implement for establishing a connection

- HTTPS – Secure HTTP
- Webservice
- REST API – REST interface required for the data reporting process
- XML – eXtensible Markup Language
- Encoding and encryption algorithms

In practice, the meaning of the above expectations is that if a piece of management software was able to establish an API connection with the Online Invoice System, it should also be able to establish a connection with the eVAT M2M system, as we do not plan to use a technology different from that of the Online Invoice System.

1.4 Technical requirements for management programs

1) The machine interface is accessible to any program (hereinafter referred to as: management program) the taxpayer intends to use that is capable of sending HTTP messages and compiling schema-compliant XML as defined in the specification to be developed in the relevant specialized system.

2) The management program must also submit the authentication details for the technical user of the taxpayer upon each data submission and data query. The implementation required for this can be defined freely by the management program. It is not necessary to create client-side automations, processes can be designed freely according to taxpayer needs.

3) To successfully perform authentication, the management program must implement the following encoding and encryption algorithms:

- BASE64 encode/decode (RFC 3548)
- SHA-512 encode (RFC 6234)
- SHA3-512 encode (FIPS 202)

The technical requirements for the management programs are very similar to those for Online Invoice API connections. The basic processes are largely identical to those established for invoice data reporting. Naturally, there are some differences, as in this case, not real-time data reporting has been developed.

2 DESCRIPTION OF API GATEWAY AUTHENTICATION

The API Gateway performs authentication and authorisation based on the data in the types described in the following sections. Each request must include the authentication details of the technical user, and the requestSignature value must be calculated for each request. The requestSignature as a signature ensures the authenticity, integrity and non-repudiation of messages.

2.1 The general structure of XML messages

For specialized systems that employ API Gateway authentication, business operations should be designed so that the request-response elements extend the BasicRequestType and BasicResponseType elements defined in common.xsd.

2.2 BasicRequestType

All request elements must have a mandatory header and user node included in the BasicRequestType. The type header contains the technical data regarding the message exchange, and the user contains the data regarding authentication.

2.2.1 BasicHeaderType

Within the requests, the header element is implemented by the BasicHeaderType element.

The structure of BasicHeaderType:

Tag	Type	Required	Content
requestId	xs:string	yes	Unique request ID
timestamp	xs:dateTime	yes	Client-side time of request (UTC)
requestVersion	xs:string	yes	Request version number
headerVersion	xs:string	no	Header version number

Facets and definitions:

Tag	SimpleType	Pattern	Enum	Default
requestId	EntityIdType	[+a-zA-Z0-9_]{1,30}	-	-
timestamp	GenericTimestampType	\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(.\d{1,3})?Z	-	-
requestVersion	AtomicStringType15	-	-	-
headerVersion	AtomicStringType15	-	-	-

Definition and related requirements

1) requestId is the identification code of the request. requestId can be of any value that matches the pattern and does not violate the uniqueness constraint. The requestId must be unique for each request, with regards to the specific taxpayer and within the timestamp tolerance. This uniqueness constraint applies to all schema-valid requests, i.e. all successful requests and all requests that are interpretable on the server side, i.e. not rejected with an INVALID_REQUEST error code. The calculated requestSignature value includes the requestId value.

The timestamp contains the client-side time that the request was submitted. The requestSignature value includes the tag value. The timestamp in the request must be received in UTC time and the proper format.

For Hungarian time zones, this translates to:

- GMT + 1 hour for DT (winter time)
- GMT + 2 hours for DST (daylight saving time)

The allowed timestamp tolerance compared to the server time is ±1 day.

For more information on dates, please see the section "**Converting local time to UTC"**.

2) requestVersion identifies the structure of the request. Subsequent interface modifications will refer back to this tag, meaning that requestVersion defines the structure of both the request and the response as well, as well as the related validations and verifications. Its value is to be filled in accordance with the value defined in the supported version. The value is checked through business validation, it has no XSD level enum value set. To be interpreted separately for each specialized system. In case of interface version changes, there may be multiple supported versions at the same time.
   - The value supported initially in the case of the eVAT system: 1.0
3) headerVersion is an optional element of the request. It serves as the tag that subsequent structures and related verifications will refer back to, should request structures undergo any fundamental changes in the future. The value is checked through business validation, it has no XSD level enum value set.

Its only currently accepted value, if it is provided: 1.0

2.2.2 UserHeaderType

Within the requests, the user element is implemented by the UserHeaderType element.

The structure of UserHeaderType:

Tag	Type	Required	Content
login	xs:string	yes	Technical user's login name
passwordHash	xs:complexType	yes	Technical user's password hash value
taxNumber	xs:string	yes	The first 8 digits of the tax number of the taxpayer who uses the interface service, and to whom the technical user is assigned
requestSignature	xs:complexType	yes	Request signature hash value

Facets and definitions

Tag	SimpleType	Pattern	Enum	Default
login	LoginType	[a-zA-Z0-9]{6,15}	-	-
passwordHash	CryptoType	-	-	-
taxNumber	TaxpayerIdType	[0-9]{8}	-	-
requestSignature	CryptoType	-	-	-

2.3 BasicResponseType

All specialized system operations have been defined so that the response element complements the BasicResponseType. The complex type and the header and result nodes defined within it are taken from common.xsd. Within the type, the header contains the transaction data for the response, and the result contains the result of processing.

The header node contained within the response will be identical to the header tags contained within the request, both in structure and in content.

In the responses, processing results are implemented by the BasicResultType element.

The structure of BasicResultType:

Tag	Type	Required	Content
funcCode	xs:string	yes	Processing result
errorCode	xs:string	no	Processing error code
message	xs:string	no	A text-based message attached to the processing result or error code
notification/notificationCode	xs:string	no	Notification code
notification/notificationText	xs:string	no	Notification text

Facets and definitions

Tag	SimpleType	Pattern	Enum	Default
funcCode	FunctionCodeType	-	OK ERROR	-
errorCode	SimpleText50NotBlankType	.*[^\s].*	-	-
message	SimpleText1024NotBlankType	.*[^\s].*	-	-
notification/ notificationCode	SimpleText100NotBlankType	.*[^\s].*	-	-
notification/notificationText	SimpleText1024NotBlankType	.*[^\s].*	-	-

Definition and related requirements

1) funcCode is the status given by the server for executing the operation contained by the request. Its interpretation may vary depending on the business operation in question, but it should always be interpreted together with the full response.
2) errorCode is returned when the value of the funcCode is ERROR. It contains the unique error code; this element can be used on the client side to map the error message. For more information on the errorCode value set, please consult the error code chart in the section "**ERROR MANAGEMENT"**.
3) message is an optional text-based message attached to the funcCode or errorCode. It is a human-readable message, to aid comprehension.
4) NAV will use the notification node to convey informational messages via API calls in the future, using a key–value structure.

2.4 Calculating requestSignature

requestSignature is a key element of the interface authentication. Its function is to prevent unauthorised persons making modifications to the system. The server side checks the hash value for each request of each operation, and only executes the operation if the correct value can be calculated from the stored and received data. requestSignature must always be an **uppercase** hash.

Due to its type, the requestSignature tag has a mandatory attribute called cryptoType.

Its only accepted value for eVAT is: **SHA3-512**

2.4.1 Calculation for file uploads

For file upload operations with the multipart/form-data media type, requestSignature can be calculated using the following method:

requestSignature is computed using the concatenation of a partial authentication and the SHA3-512 hash value of the file to be uploaded specified in the multipart/form-data request (octet-stream), and an additional SHA3-512 hash operation. The partial authentication can be derived by concatenating the following values:

- requestId value
- UTC timestamp tag value using a YYYYMMDDHHmmss mask
- string literal of the technical user's signature key

When concatenating, the date and time separators as well as the time zone must be removed for timestamp masking.

- In the case of the eVAT, the operations that have the multipart/form-data media type, and for which the requestSignature value should be calculated based on the method described above are the following:
    - manageDeclarationPartition – partition upload
    - manageAttachmentUpload – attachment upload

A fictitious example for request data:

- requestId = TSTKFT1222564
- timestamp = 2017-12-30T18:25:45.000Z
    - with masking: 20171230182545
- technical user's signature key = ce-8f5e-215119fa7dd621DLMRHRLH2S
- partial authentication value = TSTKFT1222564201712301825 45ce-8f5e-215119fa7dd621DLMRHRLH2S

- uppercase SHA3-512 hash of the octet-stream included in the request =
    - 797EB337CB3FD673976F67DE36230DFEEB3A7BC62F68423DEB3607BB211EED7E57E8515A5B8C865B97799E16961EE83FE13D5A82A4951ADF4BB42C779832883B

Thus, the base for the entire requestSignature is:

TSTKFT1222564201712301825 45ce-8f5e-215119fa7dd621DLMRHRLH2S797EB337CB3FD673976F67DE36230DFEEB3A7BC62F68423DEB3607BB211EED7E57E8515A5B8C865B97799E16961EE83FE13D5A82A4951ADF4BB42C779832883B

After SHA3-512 hashing and converting to uppercase, the value of requestSignature will be the following:

- BBC670463D11CFE8428F492807CA9086243B13015DA41605E077830EC37459543DE1C0965C2BD1A9D8811FAFAED0D465107A93D8EA0E9BBC2ECB8DCA18FB2F17

2.4.2 Calculations for other operations

As there is no file upload involved, requestSignature for any operations other than manageDeclarationPartition and manageAttachmentUpload will equal the SHA3-512 hash value for partial authentication, which can be determined by concatenating the following values:

- requestId value
- UTC timestamp tag value using a YYYYMMDDhhmmss mask
- string literal of the technical user's signature key

The SHA3-512 hash result (in capitals) of the string thus concatenated yields the requestSignature value.

2.5 General technical specifications

For services, the proper XML or multipart request must be sent in the body using the HTTP POST method, to which request the server returns an XML or multipart/form-data response. Most of the operations have the XML request-response media type; for details about the deviations from this, please refer to the interface specification of the respective specialized system.

The caller defines the operation they wish to have executed by addressing the proper endpoint and assembling a request with the appropriate structure. Depending on whether or not the request is formatted correctly, the server will either return a business XML response, or just a standard HTTP response.

In the case of eVAT:

- The definitions of common elements can be found in common.xsd and earBase.xsd. The element definitions required for communication are defined in the earApi schema definition, while the business model and element definitions for the declarations are defined in the earData schema definition.
- Exceptions to the XML request are file upload operations, which must be sent with the multipart/form-data media type: ManageDeclarationPartition, ManageAttachmentUpload
- In turn, an exception to the XML response is the QueryDeclarationData operation, which also has a multipart/form-data response.

2.5.1 HTTP headers

The request must specify the following HTTP header fields:

- content-type=application/xml
- accept=application/xml

Exceptions to this are operations with a multipart/form-data request and/or response.

The save to database and the response will always be UTF-8, regardless of the encoding specified in the inquiry, therefore it is advised to use this type of encoding in the query.

2.5.2 HTTP status codes

For correct requests, the service will always return a HTTP 200 response. This does not necessarily indicate that the business execution of the content of the request was successful, it merely indicates that the request was correctly formatted, and the resource addressed was able to read and validate it. As the error codes managed by the service are mapped, the error codes returned are also considered successful responses. This means a HTTP 200 response may also contain a message with error codes.

For an explanation of responses for incorrect requests and other technical errors, please consult the error code chart in the section **"Error management"**.

2.5.3 Size limit

The XML in the HTTP POST body sent to the service cannot be larger than 10 megabytes for operations other than file upload. Exceptions to this are file upload operations with the multipart/form-data content-type.

- When uploading a file, please refer to the eVAT interface specification for the limit currently in force.

2.5.4 Response time, timeout

The typical response time of the server is under 200 ms. The blocking timeout value for synchronous requests is 5000 ms. Please only consider response times to be timeouts on the client side if they exceed the above values. The absolute timeout value is 60 sec. Failure to respond to an operation due to a 60-second timeout does not indicate an operation failure.

2.5.5 Server clock, NTP

The server receives time settings from a closed NTP server that is not accessible to the outside world. On the client side, synchronisation with server time is not a requirement, however, optionally, the following time synchronisation is possible: http://www.pool.ntp.org/zone/hu (connection requires an NTP client).

2.5.6 Converting local time to UTC

The generation of the correct client-side requestSignature value requires that the local time be converted to UTC. This can be done by adding or subtracting from local time in the client-side time zone, the number of hours that particular time zone is offset from UTC mean time. For time zones divided into summer and winter time (daylight saving), take this into account for the addition and subtraction as well.

2.5.7 Maintenance method

In maintenance mode, the standard HTTP 503 status code indicated in the section "**Error Management"** is returned.

2.5.8 Version Control

For service-related purposes, the version is defined in the URL, while the version for the business data model is defined by the value of the requestVersion tag in the HTTP body.

Major versions are defined as the versions between which the backward compatibility of the business data model cannot be guaranteed. Minor versions are versions within a single major version where compatibility is retained for business data.

New major versions will always have a new URL and a new XML namespace. Minor versions will inherit the URL and namespace data of their major versions.

2.5.9 Character conversion

No server-side character conversion is performed on the submitted data in any case.

2.5.10 Traffic restriction

As eVAT is released as an M2M service, in the future, NAV may introduce a rate limiting solution to prevent communication that deviates from the interface specification significantly, and interferes with or hinders the operation of the system. Rate limiting means that in order to protect server-side resources, the API Gateway will be able to limit the number of requests that can be sent per taxpayer within a given time period, and if a taxpayer exceeds the number of requests set in the limit, their requests will be rejected with the HTTP standard HTTP 429 Too Many Request error code. If the limit is exceeded, the transfer of requests may be resumed after the time window has expired.

For more information on the error returned, see the section "**ERROR MANAGEMENT**".

3 ERROR MANAGEMENT

The service operates on a common list of values and error codes from the value set enumerated on the server side. Unlike the result codes, error codes intentionally do not appear in the enumerations of the schema definition, so that their potential change or expansion shall not cause implementation dependency on the client side. The result codes may be returned in the funcCode tag of the BasicResultType node, whereas the error codes may be returned in the errorCode tag in the return message. The returned funcCode values should be interpreted in accordance with the requested business process.

3.1 General error codes

3.1.1 GeneralExceptionResponseType

In every operation of the service, the response to technically unprocessable messages (improperly formatted XML, incorrect namespace or incorrect context root) will contain a GeneralExceptionResponse type error message.

This type extends BasicResultType, but contains no other element in addition to the ones contained in it.

The notification list type contains an itemised list of all schema violations related to the schema definition if the request contained at least 1 tag that is not schema valid. In this case, notificationCode = SCHEMA_VIOLATION will be set.

3.1.2 GeneralErrorResponseType

The general error type message for each operation of the service is implemented by GeneralErrorResponseType. This type extends GeneralErrorHeaderResponseType, meaning that in addition to the elements it contains, it will also incorporate a technical validation list type.

The structure of GeneralErrorResponseType:

Tag	Type	Required	Content
validationResultCode	xs:string	Yes	Result of the technical validation
validationErrorCode	xs:string	No	The technical validation error code
message	xs:string	No	Text message pertinent to the result of the technical validation

Facets and definitions

Tag	SimpleType	Pattern	Enum	Default
validationResultCode	TechnicalResultCodeType	-	CRITICAL ERROR	-
validationErrorCode	SimpleText100NotBlankType	.*[^\s].*	-	-
message	SimpleText1024NotBlankType	.*[^\s].*	-	-

Definition and related requirements

1) If a technicalValidationMessages tag is generated, the value of the validationResultCode element can only be ERROR (CRITICAL is a value reserved in this type for potential future validations).
2) The validationErrorCode tag contains the error type code.
3) The message tag contains the name and value of the incorrect tag that failed technical validation, otherwise the text-based error message associated with the validationErrorCode tag.

The technical error codes detailed in the next section are all returned in the GeneralExceptionResponse or the GeneralErrorResponse response element. The response elements detailed in the section "**Error management**" are generated only if the synchronous processing has been validated on both the technical and business logic ends. Thus, any error tag received in the HTTP response body will always indicate some kind of error.

3.2 Technical error codes

The errorCode value set of the syncronous calls is listed in the following table.

Technical and authentication errors

#	HTTP response	Response body	funcCode	errorCode	requestVersion
1	HTTP 404 NOT_FOUND	-	-	-	1.0
2	HTTP 405 METHOD_NOT_ALLOWED	GeneralExceptionResponse XML tag	ERROR	NOT_ALLOWED_EXCEPTION	1.0
3	HTTP 400 BAD_REQUEST	GeneralExceptionResponse XML tag	ERROR	INVALID_REQUEST	1.0
4	HTTP 400 BAD_REQUEST	GeneralExceptionResponse XML tag	ERROR	INVALID_REQUEST	1.0
5	HTTP 401 UNAUTHORIZED	GeneralErrorResponse XML tag	ERROR	INVALID_SECURITY_USER	1.0
6	HTTP 500 INTERNAL_SERVER_ERROR	GeneralErrorResponse XML tag	ERROR	NOT_REGISTERED_CUSTOMER	1.0
7	HTTP 500 INTERNAL_SERVER_ERROR	GeneralErrorResponse XML tag	ERROR	INVALID_CUSTOMER	1.0
8	HTTP 500 INTERNAL_SERVER_ERROR	GeneralErrorResponse XML tag	ERROR	INVALID_USER_RELATION	1.0
9	HTTP 500 INTERNAL_SERVER_ERROR	GeneralErrorResponse XML tag	ERROR	FORBIDDEN	1.0
10	HTTP 400 BAD_REQUEST	GeneralErrorResponse XML tag	ERROR	REQUEST_ID_NOT_UNIQUE	1.0
11	HTTP 400 BAD_REQUEST	GeneralErrorResponse XML tag	ERROR	INVALID_REQUEST_SIGNATURE	1.0
12	HTTP 503 SERVICE_UNAVAILABLE	GeneralErrorResponse XML tag	ERROR	SERVICE_UNAVAILABLE	1.0
13	HTTP 400 BAD_REQUEST	GeneralErrorResponse XML tag	ERROR	INVALID_TIMESTAMP	1.0
14	HTTP 400 BAD_REQUEST	GeneralErrorResponse XML tag	ERROR	INVALID_PASSWORD_HASH_CRYPTO	1.0
15	HTTP 400 BAD_REQUEST	GeneralErrorResponse XML tag	ERROR	INVALID_REQUEST_SIGNATURE_HASH_CRYPTO	1.0
16	HTTP 400 BAD_REQUEST	GeneralErrorResponse XML tag	ERROR	INVALID_REQUEST_VERSION	1.0
17	HTTP 400 BAD_REQUEST	GeneralErrorResponse XML tag	ERROR	INVALID_HEADER_VERSION	1.0

18	HTTP 415 UNSUPPORTED_MEDIA_TYPE	GeneralExceptionResponse XML tag	ERROR	INVALID_REQUEST	1.0
19	HTTP 416 NOT_ACCEPTABLE	GeneralExceptionResponse XML tag	ERROR	INVALID_REQUEST	1.0
20 *	HTTP 400 BAD_REQUEST	GeneralErrorResponse XML tag	ERROR	REQUEST_VERSION_NOT_ALLOWED	2.0
21 *	HTTP 429 TOO_MANY_REQUESTS	GeneralErrorResponse XML tag	ERROR	TOO_MANY_REQUESTS	1.0

*Error codes of validations to be introduced in the future.

Error, actions

#	errorCode	Cause of error	Action
1	-	incorrect service endpoint in request	For more information on addressing endpoints in various environments, see the section "Environment Accessibility" of the eVAT_M2M_system_interface_specification; check the URL.
2	NOT_ALLOWED_EXCEPTION	incorrect HTTP method in request	The URL is correct, but the HTTP method is not POST. All operations of the interface must be sent with POST method.
3	INVALID_REQUEST	non well-formed in XML in request body	A syntactically incorrect XML message cannot be considered XML or processed as such, as per the XML standard, and must be corrected.
4	INVALID_REQUEST	not schema-valid XML in request body	The elements of the entered XML, listed in response, violate the restrictions of the earApi.xsd, and must be corrected. (The error code may also show up if a request is not sent to the endpoint of the appropriate operation)
5	INVALID_SECURITY_USER	incorrect login + passwordHash in the request	This error message may be returned in various cases. Possible causes: no user under the entered login name, incorrect password, the login + passwordHash pair is semantically correct, but the password hash was not calculated correctly on the client side. The correctness of data and hashes should be checked; if necessary, the taxpayer using the technical user should be contacted.
6	NOT_REGISTERED_CUSTOMER	the taxpayer specified in the request is not found	No taxpayer using the system with the tax number included in the user tag can be found.
7	INVALID_CUSTOMER	incorrect taxNumber in the request	The tax number in the user tag does not exist, or its status does not allow for performing actions. The correctness of data should be checked; if necessary, the concerned taxpayer should be contacted.
8	INVALID_USER_RELATION	no connection between the entities in the request	There is no technical user with the entered login name and tax number, or the user status does not allow for performing that action any longer. The correctness of data should be checked; if necessary, the concerned taxpayer should be contacted.
9	FORBIDDEN	the technical user in the request does not have the rights to query the endpoint service	The primary users of the taxpayer allocate the rights to technical users. The concerned taxpayer should be contacted, if necessary.
10	REQUEST_ID_NOT_UNIQUE	the requestId in the request is not unique	The requestId for the tax number in the request has already been used. A new ID meeting the uniqueness requirement should be given.

11	INVALID_REQUEST_SIGNATURE	incorrect requestSignature in the request	The requestSignature calculation on the server side does not correspond with the value calculated on the client side. For more information about the calculation, see the section "Calculating requestSignature".
12	SERVICE_UNAVAILABLE	maintenance in progress	The requested operation is temporarily suspended due to maintenance. Furthermore, the error can be returned even if the Gateway or the underlying specialized system is undergoing maintenance. Please monitor the information on the interface, and repeat your request at a later time.
13	INVALID_TIMESTAMP	the timestamp indicated in the query is more than 1 day old	The error message can be returned in the operation containing all the authentications if the value of the timestamp indicated in the header node is outside the ± 1 day interval of the server time.
14	INVALID_PASSWORD_HASH_CRYPTO_TYPE	technical user password hash generation algorithm is incorrect	This error message can be returned if the value of user/passwordHash/cryptoType is not SHA-512.
15	INVALID_REQUEST_SIGNATURE_HASH_CRYPTO	the hash-generating algorithm used for signing the request is incorrect	This error message can be returned if the value of user/ requestSignature/cryptoType is not SHA3-512.
16	INVALID_REQUEST_VERSION	request version is invalid	This error message can be returned if the value of header/requestVersion is not 1.0
17	INVALID_HEADER_VERSION	header version is invalid	This error message can be returned if the value of /header/headerVersion is specified and not equal to 1.0
18	INVALID_REQUEST	the request was not submitted with the correct Content-type header	The request was not submitted with the correct media type for the operation. The request media type and/or the Content-type header value should be corrected.
19	INVALID_REQUEST	the request was not submitted with the correct Accept header	The server cannot return a response matching the Accept header in the request. The value of the Accept header needs to be corrected.
20	REQUEST_VERSION_NOT_ALLOWED	the value of the requestVersion tag in the request is no longer allowed	The requestVersion value of the request is a version that is no longer supported. This may occur when a newer version of the interface must be used due to a change in rules, where the earlier version can no longer be used from a given time. Must be corrected!
21	TOO_MANY_REQUESTS	too many requests submitted within a short time	This error message can be returned if the taxpayer has submitted too many requests within a given time window. If this is the case, requests will be rejected in order to protect resources. After the time window has expired, the transfer of requests may be resumed.

Errors in processing

#	HTTP response	Response body	funcCode	errorCode	requestVersion
1	HTTP 500 INTERNAL_SERVER_ERROR	GeneralErrorResponse	ERROR	OPERATION_FAILED	1.0

Error, actions

#	errorCode	Cause of error	Action
1	OPERATION_FAILED	unexpected processing error	Generic server-side error code. This error can occur only for synchronous requests; try repeating the operation after a short wait. If the operation is repeatedly unsuccessful in a live environment, the NAV helpdesk should be contacted; however, check the website first to make sure that there is no system outage or transient malfunction causing the error. Please note that there is no guaranteed availability in the user test environment; therefore, any errors occurring in the test system should not be reported.

4 HELPDESK AND TECHNICAL SUPPORT

This section provides information on troubleshooting and further support.

4.1 Tools assisting implementation checking

You can find more information about verifying certain codes and hashes as well as the general format of XML syntax in the following websites.

Current UTC mean time: https://www.timeanddate.com/worldclock/timezone/utc

BASE64 online encode/decode: https://www.base64decode.org/

Online CRC calculation: https://www.functions-online.com/crc32.html (online converters typically use hexadecimal values – these are acceptable, but in such cases the output must be converted to decimal value before use)

SHA-512 online encode: http://www.convertstring.com/Hash/SHA512

SHA3-512 online encode: https://codebeautify.org/sha3-512-hash-generator

Online checking of XML format and schema conformity: https://www.xmlvalidation.com/

Regex check: https://regex101.com/

XML syntax information: https://www.w3schools.com/xml/xml_syntax.asp

XML schema information: https://www.w3schools.com/xml/schema_intro.asp

4.2 Helpdesk availability

Two separate helpdesks provide support to solve problems and answer questions regarding the eVAT system. Regarding questions and problems about the live system, please send a message via https://nav.gov.hu/ugyfeliranytu/keressen_minket/levelkuldes/e-ugyfsz with the subject line ***"eÁFA, informatikai problémák" ("eVAT, IT problems")***. The form is also available in English.

Technical support for developers regarding exclusively the invoice data report interface service and exclusively in the TEST system, is available, upon request sent to the email address published on the Online Invoice System interface.

If you need technical support for using the interface, please include the content of the complete HTTP request (header and body) as well as the date of submission.

4.3 Availability on Github

4.3.1 Common repository

Repository available at: https://github.com/nav-gov-hu/Common

This repository was created for the version management of atomic types, business-catalog-type elements, and generic API communication types in a separate, common XSD (common.xsd). This allows these schema elements to be used by multiple NAV projects, thereby unifying API communication.

4.3.2 eVAT repository

eVAT repository available at: https://github.com/nav-gov-hu/eVAT