User Guide – ISIN Quest

ISIN Retrieval Tool

Version 1.63
11 Nov 2024

ISIN Quest is a utility that retrieves the reportable ISIN for a given trade date and instrument classification code.

Additional parameters may be required depending upon the type of instrument, as determined by the instrument classification code. For example, for FX Forwards, Expiry Date, Notional Currency 1, Notional Currency 2 may be required to successfully identify the exact instrument and retrieve the ISIN.

What ISIN Quest Accepts as Input Files

ISIN Quest can process input files in two formats: CSV and XML (Unsubscribed users can only upload CSV files). Regardless of the format, the file must include essential fields that are critical for querying different types of financial instruments. The specific required fields vary depending on the type of instrument. However, every input file must contain at least the following two mandatory fields:

"**instrument_classification_code**"	A 6-character Instrument Classification code. This column is used to determine which field is used for querying the data.
"**trading_date_time**"	Trading Date and Time in ISO8601 format: 2020-08-18T11:00:00.000Z, 2020-08-18 11:00:00, 2023-03-14 21:30:00+0900 (with offset). This is necessary for assessing ISIN reportability. Your input will be converted to UTC+0 time in "YYYY-MM-DD hh:mm:ss" format (e.g., 2024-01-01 12:00:00). Avoid non-standard formats like "Wed 15 Mar 2023 08:30:00 EDT" as the offset cannot be recognized and the example time will be converted to "2023-03-15 08:30:00".

Files lacking the above two fields will not be processed by ISIN Quest.

See Appendix for the columns needed for each instrument type.

The Functionality of the Tool

1. Conducts a Data Validity Check:

o Initially, ISIN Quest assesses if the input file (either CSV or XML) is in a readable format and free from malfunctions.

o For files that pass the initial check, the tool verifies if the file contains the essential field for ISIN querying, as outlined above.

2. Applies Query Rules:

o Rules are applied to each type of instrument to verify the required fields for each instrument are valid.
o If the data is valid, a query is performed to retrieve the ISIN. The retrieved ISINs are matched and displayed under a newly created column called **"ISIN"**.
o If the data is not valid for querying, comments start with "***Pre-Query:***" are provided under the newly created field called **"qomply_isin_quest_comments"**. If query was made and no reportable ISIN is given for the transaction, comments start with "***Post-Query***:" are provided for clarifications. This column provides an information as to why the data is problematic or why no reportable ISIN is found, see Appendix 3 for more information.

Result File Returned to User

Upon successful completion, the tool will return a file back to the user. The original file will be returned in a bespoke, standard format used by the platform. Three additional columns will appear in the result file, and some optional fields might as well if such fields are not given in the file:

qomply_isin_quest_comments:	If the data required for querying the ISIN for this instrument is not valid, a comment will be displayed here showing what is wrong with the data.
isin_found:	True/False indicator. If at least one ISIN based on the transaction data is found, the value will be true. For rows with invalid data (that have Pre-Query comments), no query will be made, and this field will be false.
ISIN_found_in_ANNA	The ISINs found in ANNA (if any), separated by pepe, "|" (if more than one).
ISIN	This column displays the reportable ISIN found for each transaction.
Optional Fields	Some optional fields will be used when querying the ISIN, these fields are not mandatory in the uploaded file. see Appendix 1 for more information.

Appendix

Appendix 1 - Field Requirements

The CFI codes recognized by the Tool. If you require additional CFI codes, contact support to have them added.

Optional Fields: These fields are not mandatory. If the optional fields for a specific instrument are not available in the given file, such fields will be populated with the corresponding default value. If the optional fields are given, but the field is empty for transactions, default values will also be populated. e.g. In the case of price_multiplier, the default value will be set as 1.

* For optional fields with a default value of ‘Any,’ upload transactions with or without input separately (with and without header in separate files, and upload file separately). Transactions without optional field input, but with such field header, will be treated as if the optional field received user input as ‘’ (empty string).

CFI Code Pattern	Fields Required	Optional Field
* *All CFI code* *	**- instrument_classification_code** **- trading_date_time**	**- price_multiplier = 1 (Default to 1)**
JFT**P JFR**P JFR**N HFT**P HFT**E HFM**P HFR**P	- notional_currency_1 - notional_currency_2 - expiry_date	
JFT**C JFR**C HFT**C HFM**C HFR**C	- notional_currency_1 - notional_currency_2 - expiry_date - settlement_currency	
SF***P SF***N	- isin_near_leg - isin_far_leg	
SRCCC* SRH*C*	- notional_currency_1 - notional_currency_2 - expiry_date - term_of_contract_value - term_of_contract_unit - reference_rate - reference_rate_term_unit - reference_rate_term_value	
SRCCS* SRH*S* JRIXF* SRG*S*	- notional_currency_1 - expiry_date - term_of_contract_value - term_of_contract_unit - reference_rate - reference_rate_term_unit - reference_rate_term_value	
HR**** HC****	- notional_currency_1 - expiry_date - underlying_instrument_isin	

CFI Code Pattern	Fields Required	Optional Field
HEI***	- notional_currency_1 - expiry_date - underlying_instrument_index - strike_price - strike_price_type - strike_price_currency	- underlying_instrument_index_prop = Any*
SCIC**	- notional_currency_1 - expiry_date - underlying_instrument_index - underlying_instrument_index_term_unit - underlying_instrument_index_term_value - underlying_credit_index_series - underlying_credit_index_version	- underlying_instrument_index_prop = Any*
SCUC**	- notional_currency_1 - expiry_date - underlying_instrument_isin **or** underlying_instrument_lei - debt_seniority	
SRA*C*	- notional_currency_1 - notional_currency_2 - expiry_date - term_contract_value - term_contract_unit - reference_rate - reference_rate_term_value - reference_rate_term_unit - other_leg_reference_rate - other_leg_reference_rate_term_value - other_leg_reference_rate_term_unit	
SCUTC*	- notional_currency_1 - expiry_date - underlying_instrument_isin **or** underlying_instrument_lei - debt_seniority - underlying_instrument_index - underlying_instrument_index_term_unit - underlying_instrument_index_term_value - underlying_credit_index_series - underlying_credit_index_version	- underlying_instrument_index_prop = Any*

Appendix 2 – Field Explanation

Field Name	Description
instrument_classification_code	Refers to the 6-character CFI code. This column is used to determine which field is used for querying the data, see Appendix 1 for how the first 3 characters decide which columns to be used to query the information.
expiry_date	Refers to the date on which specific security expires. It is formatted as “YYYY-MM-DD”, e.g. “2023-12-31”.
trading_date_time	Refers to the date and time of the transaction, in ISO8601 format “YYYY-MM-DDThh:mm:ss.dddddd Z”, T is a separator, ss.dddddd represents the second and its fraction of a second, and Z represents the UTC time. Similar formats such as “YYYY-MM-DD hh:mm:ss”, “YYYYMMDD hhmmss”, with or without offset, can also be accepted, and the input will be converted to UTC+0 time in “YYYY-MM-DD hh:mm:ss” format.
notional_currency_1	Refers to the currency in which the notional is denominated. It is used to indicate the notional currency of leg 1.
notional_currency_2	Refers to the second currency of the currency pair of a security
delivery_type	Refers to the method of settlement of a transaction. E.g., PHYS / CASH / OPT
option_exercise_style	Refers to the indication of whether an option may be exercised only at a fixed date (European and Asian style), a series of pre-specified dates (Bermudan), or at any time during the life of the contract (American style). Eg. EURO/AMER
price_multiplier	Optional field, default value = 1. Refers to the number of units of the underlying asset that are represented by a single derivative contract.
single_multi_currency	Refers to the currency in which a financial instrument is denominated.
contract_term	Refers to the length of time that a contract remains in effect.
underlying_instrument_code	Refers to a unique code that identifies the financial instrument which is the subject of the transaction. It is used to identify the underlying asset in a derivative contract as well as the transferable security included within article 4(1)(18©) of MiFID.
settlement_currency	Refers to the currency in which a financial instrument is settled.
Isin_near_leg	Refers to the ISIN associated with the near leg of a financial derivative contract. The “near leg” represents one of the two payment streams in the contract and typically includes cash flows scheduled to occur in the near future.
Isin_far_leg	Refers to the ISIN associated with the far leg of a financial derivative contract. The “far leg” represents the counterpart to the near leg and includes cash flows scheduled to occur in the more distant future.
notional_schedule	Refers to the predetermined set of notional amounts (or principal amounts) that are used for calculating interest payments during the life of the swap. eg. Constant/Accreting/Custom, etc.
term_of_contract_value	Works in conjunction with term_of_contract_unit to denote the duration or the length of time until the contract’s maturity. e.g., 10
term_of_contract_unit	Works in conjunction with term_of_contract_value to denote the duration or the length of time until the contract’s maturity. e.g., YEAR/MNTH
underlying_instrument_isin	Refers to the ISIN of the underlying of the contract.
underlying_instrument_index	Refers to the index associated with the underlying instrument.

Field Name	Description
underlying_instrument_index_prop	Refers to a proprietary index associated with the underlying instrument. Proprietary indices are specific to a single institution or organization.
underlying_instrument_index_term_unit	Refers to the unit of measurement associated with the term of an underlying instrument index. e.g., YEAR/MNTH
underlying_instrument_index_term_value	Refers to the numerical value associated with the term of an underlying instrument index.
underlying_credit_index_series	Refers to a series of credit indices associated with underlying financial instruments.
underlying_credit_index_version	Refers to a specific version or iteration of a credit index.
reference_rate	Refers to name of the reference rate, e.g., GBP-LIBOR, EUR-EURIBOR
reference_rate_term_unit	Works in conjunction with reference_rate_term_value to denote the duration or the length of time, e.g., MNTH
reference_rate_term_value	Works in conjunction with reference_rate_term_value to denote the duration or the length of time, e.g., 6.
other_leg_reference_rate	Refers to name of the reference rate of the other leg, e.g., GBP-LIBOR, EUR-EURIBOR
other_leg_reference_rate_term_value	Works in conjunction with other_leg_reference_rate_term_value to denote the duration or the length of time of the other leg's reference rate, e.g., MNTH
other_leg_reference_rate_term_unit	Works in conjunction with other_leg_reference_rate_term_value to denote the duration or the length of time of the other leg's reference rate, e.g., 6.
strike_price	Refers to the predetermined price at which the underlying asset of an option can be bought or sold when the option is exercised.
strike_price_type	Refers to the classification of the strike price in terms of its nature. It indicates whether the strike price is a Percentage or a Monetary Value. e.g. Percentage, Monetary Value
strike_price_currrency	Refers to the currency in which the strike price is denominated.

Appendix 3 – Comments Explained

This appendix explains the comments that may appear under the **qomply_isin_quest_comments** column in our system. These comments provide crucial insights into the status of queries related to ISIN (International Securities Identification Number) processing.

Pre-Query Comments

Comments prefixed with "Pre-Query: xxxx" indicate that a query was not executed due to data integrity issues. For example:

- **Pre-Query: Trading datetime or expiry date format is wrong**

This comment indicates you should verify the format of your trading_date_time or expiry date and attempt the query again. Any Pre-Query comments signal that the transaction has not been processed. You should verify your input based on the comment and attempt the query again.

Post-Query Comments

Comments beginning with "Post-Query: xxxx" indicate that although the query was executed, no reportable ISINs were identified for the given transaction. These comments can be classified into three types:

1. **Post-Query: No ISIN found in ANNA for this transaction**
    - This comment indicates that no ISINs matching your input were found in the ANNA database. If your input is accurate, this suggests that the transaction is not reportable.
2. **Post-Query: ISIN found in ANNA but not found in Firds**
    - This means that although ISINs corresponding to your input were located in the ANNA database, they were not found in the Firds database. As a result, the transaction is deemed not reportable. However, any ISINs found will be listed under the **ISIN_found_in_ANNA** column, separated by a vertical bar "|", this column is just for informational purposes.
3. **Post-Query: The ISINs found in Firds are not reportable for this transaction**
    - This comment indicates that while ISINs were found in both ANNA and Firds databases, the found ISINs for this transaction are not reportable. This could be due to the trading datetime of the transaction having surpassed the expiry date of the ISIN.

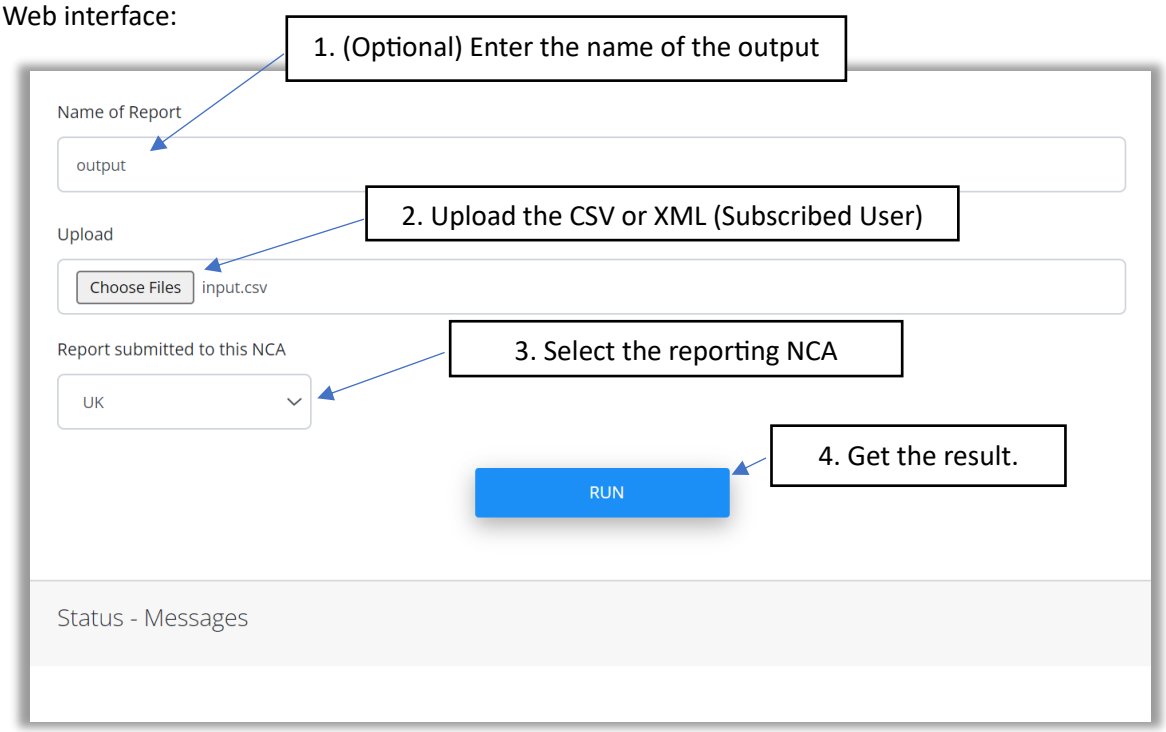
Appendix 4 – Example Use Case

Caution: The following examples are aligned for clearer demonstration, the input should be standardized CSV input, with no alignment.

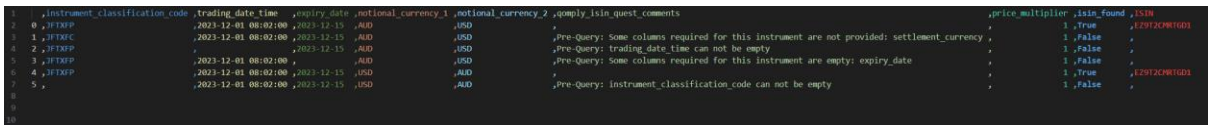
input.csv:

```
instrument_classification_code ,trading_date_time   ,expiry_date ,notional_currency_1 ,notional_currency_2
JFTXFP                         ,2023-12-01 08:02:00 ,2023-12-15  ,AUD                 ,USD
JFTXFC                         ,2023-12-01 08:02:00 ,2023-12-15  ,AUD                 ,USD
JFTXFP                         ,                    ,2023-12-15  ,AUD                 ,USD
JFTXFP                         ,2023-12-01 08:02:00 ,            ,AUD                 ,USD
JFTXFP                         ,2023-12-01 08:02:00 ,2023-12-15  ,USD                 ,AUD
                               ,2023-12-01 08:02:00 ,2023-12-15  ,USD                 ,AUD
```

Web interface:



output.csv:



Appendix 5 – ISIN Quest API Specification

Using ISIN Quest through API

The ISIN Quest API supports higher querying capacity, allows users to programmatically access ISIN Quest and enables users to integrate ISIN Quest into their own automation framework more easily.

There are three steps when using ISIN Quest API:

	Type	Description	Input	Output	Output Format
1	Authentication	Get access token	username, password	token	json
2	Request	Send a request to ISIN Quest	token, file, NCA region	submission_id	json
3	Result	Get the result	token, submission_id	result	json

Example Code:

```
# Step 1: Get Token
response = requests.post(authentication_url, data={"username": username,
                                                   "password": password})
if response.status_code == 200:
       access_token = response.json()['access']
else:
    ......

# Step 2: Send request to ISIN Quest
headers = {'Authorization': f'Bearer {access_token}'}
data = {'reportNCARegion': 2} # 2 = UK
with open(file, 'rb') as f:
    files = {'file_toupload': f}
    response = requests.post(isin_quest_request_url, data=data,
                             files=files, headers=headers)
if response.status_code == 200:
    if response.json()['is_success']:
        submission_id = response.json()['submission_id']
    else:
        ......
else:
    ......

# Step 3: Get the result
response = requests.post(isin_quest_get_result_url, headers=headers,
                         data={'submission_id': submission_id})
if response.status_code == 200:
    if json_data['is_success'] and json_data['job_finished']:
        # Success and job finished
        result = json_data['file_content']
        ......
    elif json_data['is_success'] and not json_data['job_finished']:
        # Success but job not finished
        ......
    elif not json_data['is_success']:  # No Success
        ......
else:
    ......
```

API Message body specification

```
# ISIN Quest Request Response Body
    # Satus Code 200
        # Successful
            {
                "is_success": True,
                "submission_id": xxxx,
                "error_msg": '',
                "info": '',
                "show_details": False,
            }
        # Unsuccessful
               {
                "is_success": False,
                "submission_id": 0,
                "error_msg": 'xxxxx',
                "info": '',
                "show_details": False,
                "job_finished" = True
                "job_status" = "error"
            }
    # Status code 400
               {
                "is_success": False,
                "submission_id": 0,
                "error_msg": 'xxxx,
                "info": 'Please contact Qomply support',
                "show_details": False
            }


# ISIN Quest Get Result Response
{
    "is_success": Boolean, # indicate if the run was successful
    "submission_id": int, # submission_id used to get the result
    "error_msg": string, # error msg if any
    "file_content": string, # file result as string = csv file as string
    "file_type": "csv", # format of the file here csv
    "file_separator": ";", # separator used for the csv
    "info": string, # msg to help you to fix your issue if any
    "job_finished": Boolean, # indicate if the job finished. ie the result is ready
    "job_status": string, # error/running/success
}
```