PyCEIC Python Package Developer Guide

CEIC Data – Python SDK

October 2025

ceicdata.com

Table of Contents

GETTING STARTED

OVERVIEW

The PyCEIC Python SDK provides a convenient interface for interacting with the CEIC API v2. It allows you to authenticate, request time series, search metadata, explore layouts, retrieve releases, and download insights directly in Python.

More detailed information on the CEIC API is available at:
https://developer.isimarkets.com/en/docs/ceic/macroeconomic

PYTHON SDK FEATURES

- Multiple Modules:
  - **Series** for time series and vintages
  - **Releases** for release schedules
  - **Dictionary** for metadata
  - **Layout** for data navigation
  - **Insights** for reports and dashboards
  - **Sessions** for authentication

- Automatic Code Generation: Based on CEIC's OpenAPI specification
- JSON REST Communication: All endpoints use and support JSON
- Data Science Friendly: Available Export to Pandas DataFrames for all endpoints
- Lightweight & Extensible: Easily integrated into analytics pipelines

PREREQUISITES

To use the PyCEIC SDK, you need:

- An active PyCEIC/ CEIC API subscription
- Valid CEIC login credentials or API key
- Python version 3.9+
- Pandas version 2.3.0+
- pip installed
- Network access to https://api.ceicdata.com/v2

INSTALLING AND UPDATING THE SDK

Install the package using pip:

```
pip install ceic_api_client
```

After installation, import the package in your Python scripts:

```
from ceic_api_client.pyceic import Ceic
```

AUTHENTICATION & SESSIONS API

The Sessions API handles login, logout, and environment configuration. Sessions are persisted in ~/.ceic_python_sdk/session.json and reused across calls. Only one active session is supported per user.

Ceic.login(username, password)

Authenticates a user and stores a persistent session.

Parameter	Type	Required	Default	Description
username	str	Yes	-	CEIC username
password	str	Yes	-	CEIC password

Usage Example:

```
from ceic_api_client.pyceic import Ceic
Ceic.login("user","pass")
```

Ceic.set_token()

Authenticate using an **API token** (also called API key), obtainable from your **CDMNext profile** or **API Developer Portal - Dashboard.**
This bypasses username/password and creates a persistent session using the provided token.

The token access is active until the user closes the Python IDE or the corresponding script.

Parameters

Parameter	Type	Required	Default	Description
token	str	Yes	-	CEIC API key (from your CDMNext account).

```
from ceic_api_client.pyceic import Ceic
# Set API key for all calls
token = "Your_token"
Ceic.set_token(token)
```

Ceic.logout()

Terminates the current session and clears stored credentials.

Usage Example:

```
Ceic.logout()
```

Ceic.set_region(region)

Sets the region for API calls (US or CN).

Parameter	Type	Required	Default	Description
region	str	Yes	US	Region code (US or CN)

Usage Example:

```
Ceic.set_region("US")
```

Ceic.set_environment(environment)

Sets the environment (V2 or SANDBOX).

Parameter	Type	Required	Default	Description
environment	str	Yes	V2	Environment (V2 or SANDBOX)

Usage Example:

```
Ceic.set_environment("V2")
```

Ceic.set_server(server)

Overrides the API base URL for custom configurations.

Parameter	Type	Required	Default	Description
server	str	Yes	https://api.ceicdata.com/v2	Custom API base URL

Usage Example:

```
Ceic.set_server("https://api.ceicdata.com/v2")
```

Ceic.set_proxy(proxy_url, proxy_username=None, proxy_password=None)

Configures proxy settings for the client.

Parameter	Type	Required	Default	Description
proxy_url	str	Yes	-	Proxy server URL
proxy_username	str	No	-	Proxy username
proxy_password	str	No	-	Proxy password

Usage Example:

```
Ceic.set_proxy(
proxy_url="http://proxy:8080",
proxy_username="user",
proxy_password="pass")
```

SERIES API

The Series API provides access to CEIC time series data, including metadata, values, vintages, and continuous series.

Ceic.search(keyword=None, **kwargs)

Allows searching for series by a keyword and additional filtering criteria. Each filtering criteria accepts one or more, comma-separated code values. See Dictionary functions for details on how to retrieve a specific filter code. The multi-dimensional filters include the economic classification and indicators (defined by CEIC database structure), region/country, frequency, unit, source, status and observation date. Returns an iterable object which contains result data. Each iteration will return an object with up to 100 result objects. Supports pagination and sorting.

Input Parameters:

Parameter	Type	Required	Default	Description
keyword	str	No	-	Search keyword.
frequency	list[str]	No	-	Filter by frequency (A, Q, M, W, D).
region	list[str]	No	-	Filter by region/country codes.
unit	list[str]	No	-	Filter by unit.
indicator	list[str]	No	-	Filter by indicator.
database	list[str]	No	-	Filter by database code(s).
status	list[str]	No	['T']	Series status filter: T=Active, C=Closed, B=Basic.
start_date	str	No	-	Only series with data from this date (YYYY-MM-DD) or later.
end_date	str	No	-	Only series with data up to this date (YYYY-MM-DD).
start_date_before	str	No	-	Series start date before given date.
end_date_after	str	No	-	Series end date after given date.
created_after	str	No	-	Series created after given date.
updated_after	str	No	-	Series updated after given date/time.
data_source	list[str]	No	-	Filter by data source (e.g., 'CEIC').
topic	list[str]	No	-	Filter by topic code(s).
section	list[str]	No	-	Filter by section code(s).

table	list[str]	No	-	Filter by table code(s).
filter_id	list[str]	No	-	Apply a saved filter id.
with_release_only	bool	No	False	Return only series that have a release schedule.
with_replacements_only	bool	No	False	Only series that have suggested replacements.
with_continuous_only	bool	No	False	Only series with continuous chains.
with_vintage_enabled_only	bool	No	False	Only series with vintages enabled.
subscribed_only	bool	No	False	Only series in subscribed databases.
key_only	bool	No	False	Match on 'key' series only.
name_only	bool	No	False	Match on name field only.
forecast_only	bool	No	False	Restrict to forecast series.
with_replacements_metadata	bool	No	False	Include replacements/migration metadata.
facet	list[str]	No	-	Return facets with results.
order	str	No	relevance	Sort field: relevance, updatedDate, createdDate, etc.
direction	str	No	asc	Sort direction: asc/desc.
limit	int	No	20	Results per page (max 100).
offset	int	No	0	Zero-based pagination offset.
lang	str	No	en	Language (en, zh, ru, id, jp).
format	str	No	json	Response format.
with_model_information	bool	No	False	Include API model metadata.

Output options:

- Ceic.search() → CEIC JSON-object.
- Ceic.search().as_pandas() → Pandas DataFrame object.

Output structure:

When .as_pandas() is called, the search results are normalized into a pandas.DataFrame.
The table below lists all available columns, their meaning, the **CEIC type**, and the **Pandas dtype**.

The pandas DataFrame returned by .as_pandas() also includes attributes indicating the pagination settings—such as the 'limit' and the 'total' number of search results available—to facilitate iteration through paginated data.

Output fields:

Column	Description	Type (CEIC)	Pandas dtype
id	The series unique identifier	Integer	int64
name	The series name, localized based on the lang query parameter or Accept-Language header. Defaults to English if no translated resource is available.	String	object
indicator ***(DEPRECATED)***	Please use indicators field. Contains information about the indicator including classification relation.	Indicator	object
classification ***(DEPRECATED)***	Please use indicators field. Contains information about the classification.	Classification	object
unit	Contains information about the unit.	Unit	object
country	Contains information about the country.	Country	object
province	Contains information about the province.	Province	object
frequency	Contains information about the frequency.	Frequency	object
status	Contains information about the status.	Status	object
source	Contains information about the source and the country relation.	Source	object
indicators	Array of indicator chains. Series can have multiple chains.	IndicatorsChain[]	object
geoInfo	Contains geographical information for each node.	GeoInfo[]	object
remarks	Series remarks.	String	object
mnemonic	Mnemonics information.	String	object

tradeCode	Series Trade code and Classification. Ex: `[classification]	[trade_code]`.	String
isForecast	Is series forecasted or not.	Boolean	bool
hasVintage	If true, result contains only vintage-enabled series.	Boolean	bool
headlineSeries	Show only “headline” series when set to true.	Boolean	bool
hasSchedule	If series has schedule or not.	Boolean	bool
seriesTag	The dX code of the series.	String	object
startDate	The date of the first time-point.	String (ISO 8601)	datetime64[ns]
endDate	The date of the latest time-point.	String (ISO 8601)	datetime64[ns]
multiplierCode	The series multiplier code. MN=Million, NA=No multiplier, TH=Thousand, BN=Billion, TN=Trillion, TT=Ten Thousand, QN=Quadrillion, HM=Hundred Million, HT=Hundred Thousand, 2T=Twenty Thousand, HB=Hundred Billion, TM=Ten Million, TB=Ten Billion.	String	object
lastUpdateTime	The date and time [ISO 8601] when the series was last updated in CEIC database.	String	datetime64[ns, tzutc()]
timepointsLastUpdateTime	The date and time [ISO 8601] when the timepoints series was last updated in CEIC database.	String	datetime64[ns, tzutc()]
keySeries	Indicates whether a series is a key indicator for the selected economy, as defined by CEIC analysts.	Boolean	bool
newSeries	Indicates whether a series is newly added to the database within the last 30 days.	Boolean	bool
periodEnd	Indicates the period end. • Weekly – day of week (1=Sun … 7=Sat) • Monthly – day of month (e.g. 31 → Jan 31, Feb 28/29, etc.) • Annual/quarterly/etc. – month number (1=Jan, 2=Feb, …).	Integer	int64

lastValue	Last timepoint value.	Number	float64
lastChange	Percentage change.	SeriesMetadataLastChange	object
numberOfObservations	Number of timepoints.	Integer	int64
hasContinuousSeries	If set to true, results contain only series with available historical extensions.	Boolean	bool
Replacement	Replacement IDs for disabled series	String	object
Database/ Topic/ Section/ Table	Layout information for the series	String	object

Usage Example:

```
# Search for Chinese Quarterly GDP updated after 2020. Get data
in a Dataframe
from ceic_api_client.pyceic import Ceic
results = Ceic.search(
keyword='GDP',
frequency=['Q'],
geo=[0],
updated_after='2020-01-01',
forecast_only=False,
limit=50,
order='updated',
direction='desc'
)

# CEIC JSON output
for json in results:
    print(json)

# Pandas Dataframe output
df = resu-lts.as_pandas()
print(df)
```

Ceic.series(series_id, **kwargs)

Retrieve full series details. The result contains both metadata and time-points data for one or many series IDs (max 100).

Parameter	Type	Required	Default	Description
series_id	str|list[str]	Yes	-	Series ID or list of IDs (max 100).
count	int	No	-	Number of most recent time points.
start_date	str	No	-	Filter from date (YYYY-MM-DD).
end_date	str	No	-	Filter to date (YYYY-MM-DD).
updated_after	str	No	-	Only points updated after given date/time.
blank_observations	bool	No	False	Include blank observations.
time_points_status	str	No	active	Include 'active' or 'deleted' time points.
with_replacements_ metadata	bool	No	False	Include replacements/migration metadata.
lang	str	No	en	Language.
format	str	No	json	Response format.
with_model_ information	bool	No	False	Include API model metadata.

The Ceic.series() method, when combined with .as_pandas(), returns a **pandas.DataFrame** that contains vintage economic data for the requested series IDs. This DataFrame presents tabular data normalized for easy analysis and manipulation in Python ecosystems.

General Output Options

- Ceic.series() returns the raw CEIC JSON-like object.
- Ceic.search().as_pandas() returns a standardized pandas DataFrame object.

DataFrame Structure

The returned DataFrame contains the following columns:

Column Name	Data Type	Description
last_update_date	datetime64[ns]	Date when the data was last updated
value	float64	Numeric value of the economic indicator

date	datetime64[ns]	Reference date of the observation
id	object (string)	Series ID corresponding to the economic indicator

When multiple series are queried within a single method call, the data for all series is stored collectively in a Tidy/ Long format DataFrame.

The DataFrame has appropriate datetime types for date columns and float type for numeric values, facilitating time series analysis and numerical operations.

Attributes Metadata

Alongside the tabular data, the DataFrame includes an **attributes dictionary** accessible via .attrs. This metadata contains descriptive details keyed by series ID tuples. For each series, metadata includes:

- **name**: Descriptive name of the economic indicator.
- **unit**: Measurement unit or index base reference.
- **frequency**: Data frequency, e.g., Monthly.
- **period_end**: Numeric day representing the period end (e.g., 31 for month-end).

This attribute information provides essential context for interpreting the numerical data, such as understanding the data periodicity and units.

Usage Example:

```
Ceic.series('210438802')

Ceic.series(['210438802', '370007807']) # The method supports more
than one comma-separated series

Ceic.series(['SR4478574', '370007807']) # The method supports legacy
SR Codes

Ceic.series('4d8320c9-060f-484b-ba4b-2adff3eb6f91') # The method
supports Insight series IDs

Ceic.series('370007807', start_date='2017-06-01').as_pandas() #
Filtered series information in a Pandas Dataframe
Ceic.series('370007807').as_pandas().attrs #Check attributes of the
series Timepoints dataframe

for series_result in Ceic.series([
    '371939037',
    '371938997',
    '371938957',
    '449525237',
    '481277487']
):
    for series in series_result.data:
        # Do something with the series
```

Ceic.series_metadata(series_id, **kwargs)

Retrieve only metadata for series.

Parameter	Type	Required	Default	Description
series_id	str|list	Yes	-	Series ID(s), up to 100
lang	str	No	en	Language

The Ceic.series_metadata() method, when combined with .as_pandas(), returns a **pandas.DataFrame** containing comprehensive metadata for the requested series IDs. This DataFrame serves as a normalized tabular representation of descriptive information about each economic series, designed for easy access, analysis, and reporting within Python environments.

General Output Methods

- Ceic.series_metadata() returns the raw CEIC JSON-like object with raw series metadata.
- Ceic.series_metadata().as_pandas() returns a detailed metadata DataFrame per series, describing properties, status, and structure.

Metadata DataFrame Structure

This DataFrame typically contains one row per series with the following core columns :

Column	Description	CEIC Type	Pandas dtype
id	The series unique identifier	Integer	int64
name	Localized series name based on lang parameter (defaults to English if unavailable)	String	object
indicators	Array of indicator chains describing classification relations	IndicatorsChain []	object
indicator (deprecated)	Use indicators instead. Contains classification info	Indicator	object
classification (deprecated)	Use indicators instead. Contains classification info	Classification	object
unit	Unit of measurement or index base	Unit	object
country	Country associated with the series	Country	object
province	Province or region info	Province	object
frequency	Data frequency (e.g., Monthly)	Frequency	object

status	Status of the series (e.g., Active)	Status	object
source	Data source and country relation	Source	object
geo_info	Geographical info for each node	GeoInfo[]	object
remarks	Additional remarks or notes	String	object
mnemonic	Mnemonics used for series shortcuts	String	object
is_forecast	Boolean indicating if series data is forecasted	Boolean	bool
has_vintage	Boolean indicating series supports vintage data	Boolean	bool
headline_series	Boolean indicating if series is a headline indicator	Boolean	bool
has_schedule	Boolean indicating presence of release schedule	Boolean	bool
series_tag	Code identifying series grouping	String	object
start_date	Date of first time-point	ISO 8601 String	datetime64[ns]
end_date	Date of most recent time-point	ISO 8601 String	datetime64[ns]
multiplier_code	Multiplier scale code (e.g., MN=Million, BN=Billion)	String	object
last_update_date	Date and time (UTC) when series was last updated	ISO 8601 String	datetime64[ns, tzutc()]
time_points_last_update_time	Date and time (UTC) when time points were last updated	ISO 8601 String	datetime64[ns, tzutc()]
key_series	Boolean indicating key economic indicator	Boolean	bool
new_series	Boolean indicating if the series has been recently added	Boolean	bool
period_end	Numeric indicator of period end (e.g., day of month or month number)	Integer	int64
last_value	Last available data point value	Number	float64
last_change	Percentage change of last value	SeriesMetadata LastChange	object

number_of_observations	Number of data points (time points)	Integer	int64
has_continuous_series	Boolean indicating if series has continuous historical data	Boolean	bool
trade_code	Trade code classification info	String	object
replacement	Replacement series ID for disabled data	String	object

Usage Example:

```
Ceic.series_metadata('210438802')

Ceic.series_metadata(['210438802', '370007807']) # The method
supports more than one comma-separated series

Ceic.series_metadata(['SR347217', '370007807']) # The method supports
legacy SR Codes

Ceic.series_metadata('4d8320c9-060f-484b-ba4b-2adff3eb6f91') # The
method supports Insight series IDs

Ceic.series_metadata('370007807').as_pandas() # series information in
a Pandas Dataframe
```

Ceic.series_data(series_id, **kwargs)

Retrieve time points only for one or many series IDs (max 100).

Parameter	Type	Required	Default	Description
series_id	str|list[str]	Yes	-	Series ID(s).
count	int	No	-	Number of most recent time points.
start_date	str	No	-	Start date (YYYY-MM-DD).
end_date	str	No	-	End date (YYYY-MM-DD).
vintage_revision_date	str	No	-	Return snapshot at vintage date (YYYY-MM-DD).
updated_after	str	No	-	Only points updated after given date/time.
blank_observations	bool	No	False	Include blank observations.
time_points_status	str	No	active	Include 'active' or 'deleted' time points.
lang	str	No	en	Language.
format	str	No	json	Response format.
with_model_information	bool	No	False	Include API model metadata.

The Ceic.series_data() method, when combined with .as_pandas(), returns a **pandas.DataFrame** containing timepoint observations (data points) for the requested series IDs. This DataFrame offers a tidy/long format dataset designed for efficient time series analysis and visualization within Python.

General Output Methods

- Ceic.series_data() returns raw timepoints JSON-like objects.
- Ceic.series_data().as_pandas() returns a slim DataFrame containing only timepoint data (observations) without metadata in attributes.

DataFrame Structure

The returned DataFrame contains the following columns (based on a sample of 2340 rows and 4 columns):

Column	Description	Pandas dtype
last_update_date	Date when this particular value was last updated	datetime64[ns]
value	Numeric observation value	float64
date	Reference date of the observation	datetime64[ns]
id	Series ID corresponding to the observation	object (string)

- Columns representing timestamps use appropriate datetime types for time-based operations.
- value is a float numeric type supporting quantitative analysis.
- The Series id provides linkage back to the series metadata and data provenance.

Usage Example:

```
Ceic.series_data('210438802')

Ceic.series_data(['210438802', '370007807']) # The method
supports more than one comma-separated series
Ceic.series_data(['SR4478574', '370007807']) # The method
supports legacy SR Codes

Ceic.series_data('4d8320c9-060f-484b-ba4b-2adff3eb6f91') # The
method supports Insight series IDs

Ceic.series_data('370007807', start_date='2017-06-
01').as_pandas() # Filtered series information in a Pandas
Dataframe

for series_result in
Ceic.series_data(['371939037','371938997','371938957','449525237
','481277487']):
    for series in series_result.data:
        # Do something with the series
```

Ceic.series_layouts(series_id, **kwargs)

Retrieve layout information for a series (database/topic/section ancestry).

The method supports Pandas Dataframe output using .as_pandas()

Parameter	Type	Required	Default	Description
series_id	str|list	Yes	-	Series ID(s)

Usage Example:

```
Ceic.series_layouts(["3775101","210438802"]) #Retrieve all Layouts for a series/ list of series
Ceic.series_layouts(["3775101","210438802"]).as_pandas() # Retrieve all Layouts for a series/ list of series as a Pandas dataframe
```

Ceic.series_statistics(series_id, **kwargs)

Retrieve summary statistics for series (Variance, Std Dev, Skewness, Kurtosis, Min, Max, Mean, etc.).

The method supports Pandas Dataframe output using .as_pandas()

Parameter	Type	Required	Default	Description
series_id	str|list	Yes	-	Series ID(s), up to 100

Usage Example:

```
Ceic.series_statistics("3775101") #Retrieve all series statistics for a single series
Ceic.series_statistics(["3775101","210438802"]) #Retrieve all series statistics for multiple series
Ceic.series_statistics(["3775101","210438802"]).as_pandas() # Retrieve all series statistics for a series/ list of series as a Pandas dataframe
```

Ceic.series_continuous_info(series_id, **kwargs)

Retrieve information about continuous series (linked historical discontinued/ rebased data). Continuous series is regulated with the **'hasContinuousSeries' = TRUE** tag.

The method extends the history and fill gaps of active series by combining older and newer counterparts of comparable time series to generate a longer series.

Depending on the dataset, data history is extended, and gaps are filled using a combination of geometric compounding and/or filling the missing time points with the comparable older/newer series. CEIC calculates geometric compounding by using the ratio of the first observation of the newer series and the corresponding overlapping time point from the older series, subsequently adjusting observations from the older series using this ratio (i.e., the linking factor). Series estimated using geometric compounding are rounded, if applicable.

The method returns function description, chain_id and series IDs in the historical extension.

The method supports **Pandas Dataframe** output using .as_pandas()

Parameter	Type	Required	Default	Description
series_id	str	Yes	-	Series ID

Usage Example:

```
Ceic.series_continuous_info("386587797") #Retrieve continuous
series information
Ceic.series_continuous_info("386587797").as_pandas() #Retrieve
continuous series information in a Pandas Dataframe format
```

Ceic.series_continuous_data(series_id, chain_id, **kwargs)

Retrieve time-points for a specific continuous chain. The method returns continuous series timepoints with applied functions. The method supports **Pandas Dataframe** output using .as_pandas()

Parameter	Type	Required	Default	Description
series_id	str	Yes	-	Series ID
chain_id	str	Yes	-	Continuous chain ID

Usage Example:

```
Ceic.series_continuous_data("386587797", chain_id = "1018864")
#Retrieve Spliced series data
Ceic.series_continuous_data("386587797", chain_id =
"1018864").as_pandas() #Retrieve Spliced series data in a Pandas
Series object
```

POINT-IN-TIME Methods

The following methods are available upon subscription to the Point-in-Time database and available for series tagged with **'vintage_enabled' = TRUE**.

Ceic.series_vintages(series_id, **kwargs)

The Ceic.series_vintages() method retrieves all vintage timepoint data (historical revisions) for one or more economic series. This method supports returning the data as a pandas.DataFrame by using .as_pandas().

Parameter	Type	Required	Default	Description
series_id	str|list	Yes	-	Series ID(s), up to 100
vintages_start_date	str	No	-	Start date for vintages
vintages_end_date	str	No	-	End date for vintages
vintages_count	int	No	12	Number of vintages to return
count	int	No	-	Number of time-points per vintage

Dataframe Structure:

Column	Description	Pandas dtype
revision_date	The date when the vintage revision was recorded	datetime64[ns, UTC]
vintage_value	The value of the series at that vintage	float64
date	The reference date for the vintage value	datetime64[ns]
id	Series ID associated with the vintage data	object (string)

Usage Example:

```
result = Ceic.series_vintages("5793901")
df = result.as_pandas()
df.head(50)
```

Ceic.series_vintages_dates(series_id, **kwargs)

Retrieve all vintage release dates.

Parameter	Type	Required	Default	Description
series_id	str|list	Yes	-	Series ID(s), up to 100
vintages_start_date	str	No	-	Start date for vintages
vintages_end_date	str	No	-	End date for vintages
vintages_count	int	No	12	Number of vintages to return
count	int	No	-	Number of time-points per vintage

Usage Example:

```
Ceic.series_vintages_dates("5793901")
```

Ceic.series_earliest(series_id, **kwargs)

Retrieves the earliest available vintage for each timepoint for one or more series IDs or codes. Supports **Pandas DataFrame** output using .as_pandas(). Returns results in a tidy, long-format DataFrame when multiple series are requested.

Parameter	Type	Required	Default	Description
series_id	str|list	Yes	-	Series ID(s)

Dataframe Structure:

Column	Description	Pandas dtype
Last_update_date	The date when the vintage revision was recorded	datetime64[ns, UTC]
value	The value of the series at that vintage	float64
date	The reference date for the vintage value	datetime64[ns]
id	Series ID associated with the vintage data	object (string)

Usage Example:

```
Ceic.series_earliest("5793901") #Access the earliest available
vintage per timepoint
```

```
Ceic.series_earliest("5793901").as_pandas() # Get a Pandas
dataframe/ series direct output
```

Ceic.series_vintages_as_dict(series_id, **kwargs)

Returns a full vintage/revision matrix for a series, with rows as observation dates, columns as vintage dates (publication dates), and cell values as the series values available at that vintage. Missing values (NaNs) are filled forward with the latest available data to fully populate the matrix.

Note: Generating the vintage matrix may be time-consuming depending on the size of the resulting matrix.

Parameter	Type	Required	Default	Description
series_id	str	Yes	-	Series ID
vintages_start_date	str	No	-	Start date for vintages
vintages_end_date	str	No	-	End date for vintages
vintages_count	int	No	12	Number of vintages to return
count	int	No	-	Number of time-points per vintage

Usage Example:

```
data = Ceic.series_vintages_as_dict(41091101)
df = pd.DataFrame(data)
print(df.head(50))
```

Ceic.series_vintages_continuous(series_id, **kwargs)

Retrieve information about linked vintage series (linked historical chains of vintages). Depending on the dataset, data history is extended, and gaps are filled using a combination of geometric compounding and/or filling the missing time points with the comparable older/newer series. CEIC calculates geometric compounding by using the ratio of the first observation of the newer series and the corresponding overlapping time point from the older series, subsequently adjusting observations from the older series using this ratio (i.e., the linking factor). Series estimated using geometric compounding are rounded, if applicable.

The method returns function description, chain_id and series IDs in the historical extension.

This method supports output as a **Pandas DataFrame** via the .as_pandas() method.

Parameter	Type	Required	Default	Description
series_id	str|list	Yes	-	Series ID(s), up to 100

Usage Example:

```
Ceic.series_vintages_continuous("365749427") #Retrieve
continuous series information for vintages

Ceic.series_vintages_continuous("365749427").as_pandas()
#Retrieve continuous series information for vintages in a Pandas
Dataframe format
```

Ceic.series_vintages_continuous_chain(series_id, chain_id, **kwargs)

Retrieve timepoints of linked vintage series (linked historical chains of vintages).

Parameter	Type	Required	Default	Description
series_id	str	Yes	-	Series ID
chain_id	str	Yes	-	Continuous chain ID

Usage Example:

```
Ceic.series_vintages_continuous_chain("365749427", chain_id =
"1502085") #Retrieve Spliced series data
Ceic.series_vintages_continuous_chain("365749427", chain_id =
"1502085").as_pandas() #Retrieve Spliced series data in a Pandas
Series object
```

RELEASES API

The Releases API provides access to release schedules and their associated series. The Releases methods are available for series tagged with **'hasSchedule'= TRUE**

Ceic.releases(keyword=None, **kwargs)

Search release schedules by keyword and filters. This method supports output as a Pandas DataFrame via the .as_pandas() method.

Input parameters

Parameter	Type	Required	Default	Description
keyword	str	No	-	Search keyword.
release_date_from	str	No	-	Filter releases from (YYYY-MM-DD).
release_date_to	str	No	-	Filter releases to (YYYY-MM-DD).
release_status	str	No	Released	Status (Pending, Delayed, Released).
order	str	No	releaseDate	Sort field.
direction	str	No	desc	Sort direction (asc/desc).
limit	int	No	20	Results per page (max 100).
offset	int	No	0	Pagination offset.
lang	str	No	en	Language.
format	str	No	json	Response format.
with_model_information	bool	No	False	Include API model metadata.

Usage Example:

```
# Retrieve upcoming releases related to 'US
releases = Ceic.releases(
    keyword='US GDP',
    limit=20,
    direction='asc',
    release_status=['Pending', 'Delayed'],
    release_date_from='2025-09-01',
    release_date_to='2025-12-01'
)
# Convert to pandas DataFrame if needed:
releases_df = releases.as_pandas()
# Preview results
print(releases_df.head())
```

Ceic.release_series(code, **kwargs)

Retrieve all series associated with a specific release code.

Parameter	Type	Required	Default	Description
code	str	Yes	-	Release code identifier.
limit	int	No	20	Results per page (max 100).
offset	int	No	0	Pagination offset.
lang	str	No	en	Language.
format	str	No	json	Response format.
with_model_information	bool	No	False	Include API model metadata.

Usage example:

```
Ceic.release_series(code='50058307')
```

Ceic.series_releases(series_id, **kwargs)

Retrieve release schedule for a specific series. This method supports output as a Pandas DataFrame via the .as_pandas() method.

Parameter	Type	Required	Default	Description
series_id	str	Yes	-	Series ID.
release_date_from	str	No	-	Filter from date.
release_date_to	str	No	-	Filter to date.
release_status	str	No	-	Status filter.
limit	int	No	20	Results per page (max 100).
offset	int	No	0	Pagination offset.
lang	str	No	en	Language.
format	str	No	json	Response format.
with_model_information	bool	No	False	Include API model metadata.

Usage example:

```
# Retrieve series releases for a given series ID and access the
output in a pd Dataframe
df_releases =
Ceic.series_releases(series_id="5823501").as_pandas()
print(df_releases.head())
```

DICTIONARIES API

The Dictionaries API provides metadata used for searching and filtering series: indicators, classifications, regions/geo, sources, units, frequencies, and statuses. Results can be converted to pandas DataFrames via .as_pandas().

Ceic.dictionaries(**kwargs)

Returns a list of available dictionary endpoints and metadata. This method supports output as a Pandas DataFrame via the .as_pandas() method.

Parameter	Type	Required	Default	Description
lang	str	No	en	Language of results
format	str	No	json	Output format (json, xml, csv)
with_model_information	bool	No	False	Include API model metadata

Usage Example:

```
Ceic.dictionaries().as_pandas() #retrieve a Dataframe output
Ceic.dictionaries() #retrieve a JSON output
```

Ceic.indicators(**kwargs)

Returns the list of indicators across datasets (used to filter/search series).

Parameter	Type	Required	Default	Description
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
inds = Ceic.indicators(limit=50)
inds_df = inds.as_pandas()
print(inds_df)
```

Ceic.classifications(**kwargs)

DEPRECATED. Returns available indicator classifications (top-level taxonomies).

Parameter	Type	Required	Default	Description
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
cl = Ceic.classifications()
cl_df = cl.as_pandas()
```

Ceic.classification_indicators(classification_id, **kwargs)

DEPRECATED. Returns all indicators within a given classification.

Parameter	Type	Required	Default	Description
classification_id	str	Yes	-	Classification code or ID
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
Ceic.classification_indicators(classification_id='200030')
```

Ceic.geo(**kwargs)

Returns geographical entities (countries, regions, provinces) used in series metadata.

Parameter	Type	Required	Default	Description
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
Ceic.geo()
```

Ceic.country_sources(country_id, **kwargs)

Returns sources available for a given country/region code.

Parameter	Type	Required	Default	Description
country_id	str	Yes	-	Country or region code (e.g., 'US')
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
cs = Ceic.country_sources("US")
cs_df = cs.as_pandas()
```

Ceic.regions(**kwargs)

Returns a list of regions (macro/geo groupings).

Parameter	Type	Required	Default	Description
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
regs = Ceic.regions()
regs_df = regs.as_pandas()
```

Ceic.sources(**kwargs)

Returns a list of data sources (national statistics offices, central banks, etc.).

Parameter	Type	Required	Default	Description
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
src = Ceic.sources(limit=500)
src_df = src.as_pandas()
```

Ceic.units(**kwargs)

Returns measurement units (%, USD, index points, etc.).

Parameter	Type	Required	Default	Description
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
units = Ceic.units()
units_df = units.as_pandas()
```

Ceic.frequencies(**kwargs)

Returns available series frequencies (A, Q, M, W, D, etc.).

Parameter	Type	Required	Default	Description
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
freqs = Ceic.frequencies()
```

```
freqs_df = freqs.as_pandas()
```

Ceic.statuses(**kwargs)

Returns series statuses (Active, Closed, Basic).

Parameter	Type	Required	Default	Description
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
sts = Ceic.statuses()
sts_df = sts.as_pandas()
```

LAYOUTS API

The Layouts API allows navigation of the CEIC hierarchical structure: Databases → Topics → Sections → Tables → Series. It is useful for discovering available data within CEIC.

Ceic.layout_databases(**kwargs)

Returns a list of available databases.

Parameter	Type	Required	Default	Description
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
from ceic_api_client.pyceic import Ceic
dbs = Ceic.layout_databases()
dbs_df = dbs.as_pandas()
```

Ceic.layout_database_topics(database_id, **kwargs)

Returns topics for a given database.

Parameter	Type	Required	Default	Description
database_id	str	Yes	-	Database code or ID
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
topics = Ceic.layout_database_topics("DB_CODE")
topics_df = topics.as_pandas()
```

Ceic.layout_topic_sections(topic_id, **kwargs)

Returns sections for a given topic.

Parameter	Type	Required	Default	Description
topic_id	str	Yes	-	Topic code or ID
lang	str	No	en	Language
format	str	No	json	Output format

limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
sections = Ceic.layout_topic_sections("TOPIC_CODE")
sections_df = sections.as_pandas()
```

Ceic.layout_section_tables(section_id, **kwargs)

Returns tables for a given section.

Parameter	Type	Required	Default	Description
section_id	str	Yes	-	Section code or ID
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
tables = Ceic.layout_section_tables("SECTION_CODE")
tables_df = tables.as_pandas()
```

Ceic.layout_table_series(table_id, **kwargs)

Returns series for a given table.

Parameter	Type	Required	Default	Description
table_id	str	Yes	-	Table code or ID
lang	str	No	en	Language
format	str	No	json	Output format
limit	int	No	100	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
series = Ceic.layout_table_series("TABLE_CODE")
series_df = series.as_pandas()
```

Ceic.footnotes(node_code, download_path, **kwargs)

Downloads footnotes for a given node (topic, section, or table) to a local path.

Parameter	Type	Required	Default	Description
node_code	str	Yes	-	Node identifier
download_path	str	Yes	-	Local path to save footnotes
lang	str	No	en	Language

Usage Example:

```
Ceic.footnotes(node_code="WORLD_GDP_TABLE",
download_path="./footnotes")
```

INSIGHTS API

The Insights API provides access to CEIC Insights: curated dashboards and reports created by CEIC analysts or users. It allows searching, retrieving metadata, downloading reports, and accessing series within insights.

Ceic.insights(**kwargs)

Returns a list of all insights available to the user.

Parameter	Type	Required	Default	Description
lang	str	No	en	Language
format	str	No	json	Output format
with_model_information	bool	No	False	Include API model metadata
limit	int	No	20	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
from ceic_api_client.pyceic import Ceic
ins = Ceic.insights(limit=50)
df = ins.as_pandas()
```

Ceic.search_insights(**kwargs)

Search insights by group, tags, and categories.

Parameter	Type	Required	Default	Description
group	str	No	all	Group filter (favorite, my, analytics, shared, recent, gallery, data_talk, etc.)
tags	list	No	-	Filter by tags
categories	list	No	-	Filter by categories
order	str	No	relevance	Sort order
direction	str	No	desc	Sort direction
limit	int	No	20	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
ins = Ceic.search_insights(group="analytics",
tags=["inflation"], limit=20)
df = ins.as_pandas()
```

Ceic.insights_categories(**kwargs)

Retrieve categories available for insights groups.

Parameter	Type	Required	Default	Description
group	str	Yes	-	Group type (favorite, my, shared, recent, all)
lang	str	No	en	Language

Usage Example:

```
cats = Ceic.insights_categories(group="my")
cats_df = cats.as_pandas()
```

Ceic.gallery_insights_categories(**kwargs)

Retrieve categories for gallery-type insights (analytics, gallery).

Parameter	Type	Required	Default	Description
group	str	Yes	-	Group type (analytics, gallery)
lang	str	No	en	Language

Usage Example:

```
cats = Ceic.gallery_insights_categories(group="analytics")
cats_df = cats.as_pandas()
```

Ceic.insight(insight_id, **kwargs)

Retrieve details of a specific insight.

Parameter	Type	Required	Default	Description
insight_id	str	Yes	-	Insight ID
lang	str	No	en	Language

Usage Example:

```
info = Ceic.insight("INSIGHT_ABC")
info_df = info.as_pandas()
```

Ceic.download_insight(insight_id, file_format, **kwargs)

Download an insight report as XLSX or PDF. Returns short-lived download links (~5 min expiry).

Parameter	Type	Required	Default	Description
insight_id	str	Yes	-	Insight ID
file_format	str	Yes	-	Report format (xlsx, pdf)

Usage Example:

```
dl = Ceic.download_insight("INSIGHT_ABC", "xlsx")
print(dl) # contains one or more URLs
```

Ceic.insight_series(insight_id, **kwargs)

Retrieve all series contained in an insight.

Parameter	Type	Required	Default	Description
insight_id	str	Yes	-	Insight ID
count	int	No	-	Number of time-points per series
start_date	str	No	-	Start date filter
end_date	str	No	-	End date filter
updated_after	str	No	-	Return updated after date
blank_observations	bool	No	False	Include blanks
time_points_status	str	No	active	Filter by status
limit	int	No	20	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
isr = Ceic.insight_series("INSIGHT_ABC", count=12)
```

```
df = isr.as_pandas()
```

Ceic.insight_series_data(insight_id, **kwargs)

Retrieve time-points for all series within an insight.

Parameter	Type	Required	Default	Description
insight_id	str	Yes	-	Insight ID
count	int	No	-	Number of time-points per series
start_date	str	No	-	Start date filter
end_date	str	No	-	End date filter
updated_after	str	No	-	Return updated after date
blank_observations	bool	No	False	Include blanks
time_points_status	str	No	active	Filter by status
limit	int	No	20	Results per page
offset	int	No	0	Pagination offset

Usage Example:

```
isd = Ceic.insight_series_data("INSIGHT_ABC", count=24)
df = isd.as_pandas()
```

USAGE EXAMPLES

This section provides practical examples that combine multiple API endpoints to solve real-world data tasks.

Example 1: Paginated Search Data Retrieval Workflow

```
# Search for US non farm Payroll Monthly data and iterate
through all results to create a single dataframe

import pandas as pd
import time

# Initial search to get batch size and total results
ceic_search = Ceic.search(
    "US non farm payroll",
    frequency=["M"],
    status=["T", "C", "B"],
    order="popular",
    direction="desc",
    lang="en",
    offset=0, # start at 0
    limit=100 # max batch size
)

batch_size = ceic_search.attrs['batch_size']
total = ceic_search.attrs['total']

all_results = []
offset = 0

while offset < total:
    page_df = Ceic.search(
        "US non farm payroll",
        frequency=["M"],
        status=["T", "C", "B"],
        order="popular",
```

```
            direction="desc",
            lang="en",
            offset=offset,
            limit=batch_size
        ).as_pandas()

    start = offset
    end = min(offset + batch_size - 1, total - 1)
    print(f"Search results from {start} to {end}")

    all_results.append(page_df)
    offset += batch_size

    time.sleep(0.3)

    print("Pagination loop complete.")
    df_all = pd.concat(all_results, ignore_index=True)
    print(f"Total number of series loaded into the dataframe:
    {len(df_all)}")
```

Example 2: Use Ceic.series() with multiple series and plot using the series name from the attributes

```
import matplotlib.pyplot as plt

# Perform search for only the first 3 series IDs (example IDs)
series_ids = ["5724301", "5793201", "143227301"]
ceic_result_df = Ceic.series(series_ids, lang="en").as_pandas()

# Extract attributes dictionary for labels
attrs = ceic_result_df.attrs

# Extract unique IDs from the DataFrame (automatically from the
search)
unique_ids = ceic_result_df['id'].unique()
plt.figure(figsize=(14, 7))
```

```
    for sid in unique_ids:
        # Filter and sort data for each series
        series_data = ceic_result_df[ceic_result_df['id'] ==
        sid].sort_values('date')

        # Obtain descriptive name from attributes; fallback to ID
        string
        sid_key = f"({sid},)"
        attr = attrs.get(sid_key, {})
        series_name = attr.get('name', (f"Series {sid}",))[0]

        plt.plot(series_data['date'], series_data['value'],
        label=series_name)

plt.title("CEIC Economic Series Over Time - First 3 Series")
plt.xlabel("Date")
plt.ylabel("Value")
plt.legend() plt.grid(True)
plt.tight_layout()
plt.show()
```

Example 3: Search for GDP data for China and plot the series

1. Search for US quarterly GDP series
2. Fetch time-points
3. Convert to pandas DataFrame
4. Plot using matplotlib

```
from ceic_api_client.pyceic
import Ceic import matplotlib.pyplot as plt

Ceic.login("user", "pass")
result = Ceic.search("GDP", frequency=["Q"], region=["US"])

sid = result.as_pandas().iloc[0]['id']
series = Ceic.series_data(sid, count=40)
df = series.as_pandas() df.plot(x="Date", y="Value", title="US
GDP Quarterly")
plt.show()
Ceic.logout()
```

Example 4: Monitor CPI Release Calendar

Use the Releases API to track upcoming CPI releases.

```
rels = Ceic.releases("CPI", release_date_from="2024-01-01")
df = rels.as_pandas()
print(df[['releaseDate','name']])
```

Example 5: Download an Insight Report

Search for an Insight, then download it as Excel.

```
ins = Ceic.search_insights(group="analytics", tags=["inflation"])
iid = ins.as_pandas().iloc[0]['id']
dl = Ceic.download_insight(iid, "xlsx")
print(dl)
```

ERROR HANDLING & BEST PRACTICES

Common Exceptions:

- CeicInvalidLoginDetailsException → Invalid username/password
- CeicSessionExpiredException → Session has expired
- CeicInvalidInputParameterException → Wrong parameter value
- CeicNoActiveSessionsException → Logout attempted without login

Best Practices:

- Batch requests up to 100 IDs for efficiency
- Use updated_after to fetch only recently updated series
- Cache dictionary lookups (countries, indicators) locally
- Handle retries gracefully (API may throttle)
- Always logout after session to clean up

TROUBLESHOOTING

Common issues and resolutions:

- **401 Unauthorized** → Check login credentials or API key
- **SSL Errors** → Set config.verify_ssl = False (not recommended)
- **Proxy issues** → Configure via Ceic.set_proxy()
- **Insight download expired** → Re-run Ceic.download_insight within 5 min
- **PackageUpdateWarning** → Update SDK to the latest version

SUPPORT & CONTACT

For technical support, please contact CEIC Helpdesk:
📧 helpdesk@ceicdata.com

You can also access the CEIC Documentation Portal:
https://developer.isimarkets.com/en/docs/ceic/macroeconomic

For CDMNext Help, visit:
https://insights.ceicdata.com/login

APPENDIX – FUNCTION REFERENCE INDEX

This appendix provides a quick reference index of all PyCEIC facade functions.

- Authentication: Ceic.login, Ceic.logout, Ceic.set_token, Ceic.set_region, Ceic.set_environment, Ceic.set_server, Ceic.set_proxy

- Series: Ceic.series, Ceic.series_vintages, Ceic.series_metadata, Ceic.series_data, Ceic.series_earliest, Ceic.series_layouts, Ceic.series_statistics, Ceic.series_continuous_info, Ceic.series_continuous_data, Ceic.search

- Releases: Ceic.releases, Ceic.release_series, Ceic.series_releases

- Dictionaries: Ceic.dictionaries, Ceic.indicators, Ceic.classifications, Ceic.classification_indicators, Ceic.geo, Ceic.country_sources, Ceic.regions, Ceic.sources, Ceic.units, Ceic.frequencies, Ceic.statuses

- Layouts: Ceic.layout_databases, Ceic.layout_database_topics, Ceic.layout_topic_sections, Ceic.layout_section_tables, Ceic.layout_table_series, Ceic.footnotes

- Insights: Ceic.insights, Ceic.search_insights, Ceic.insights_categories, Ceic.gallery_insights_categories, Ceic.insight, Ceic.download_insight, Ceic.insight_series, Ceic.insight_series_data