REFINITIV

TICK HISTORY

BEST PRACTICES & LIMITS

DOCUMENT VERSION 1.2

Date of Issue: July 2020

.

Contents

About This Document

The *Tick History Best Practices and Limits* guide describes recognized techniques for using Tick History more quickly and effectively while reducing opportunities for error. It also describes Tick History limits you will want to keep in mind when issuing report requests. Some practices and limits apply to using Tick History via its graphical user interface (GUI), some to using Tick History via its REST API, and some to both interfaces.

This guide is intended for all Tick History users and developers.

Feedback

We invite your comments, corrections, and suggestions about this document: access the Feedback option under Help & Support at MyRefinitiv. Your feedback helps us continue to improve our user assistance.

Support

The Refinitiv Statement of Service is available on MyRefinitiv. MyRefinitiv is the Refinitiv portal that provides a single access point for timesaving support services, along with billing, user management, and information. For support using Tick History, please raise a query by accessing Help & Support at MyRefinitiv.

You are encouraged to subscribe to the following support channels to keep informed of changes to products and data, and to be notified of any service issues or changes:

- **Change Notifications**
  - **Product** change notifications detail new, enhanced, or changed functionality, which may require your action, in products that you use.
  - **Content** change notifications alert you to upcoming changes to real-time and historical data across all asset classes that are relevant to you.
  - **RIC** change notifications inform you of planned changes to Reuters Instrument Codes.
- **Service Alerts**

  You can subscribe to alerts about planned maintenance and unplanned service issues affecting your products and services, and be notified via SMS or email.

Your Personal Information

Refinitiv is committed to the responsible handling and protection of personal information. We invite you to review our Privacy Statement, which describes how we collect, use, disclose, transfer, and store personal information when needed to provide our services and for our operational and business purposes.

1 Best Practices Common to All Interfaces

This chapter describes Tick History best practices that apply to all interfaces (GUI and REST API):

- Specify Only the Dates, Fields, and Instruments You Need
- Querying Multiple Instruments Together Is Faster Than One at a Time
- Use a URL Based on Domain Name, Not on IP Address
- Get Output Files Faster by Downloading Them Directly From Amazon
- Retrieve Reports When Completed, Not in the Sequence Submitted
- Download Reports before They Become Unavailable
- Use Extraction Notes to Troubleshoot Reports
- Avoid Open-Ended Searches
- Java Support

Specify Only the Dates, Fields, and Instruments You Need

Report performance can be very sensitive to the length of the date range on which you are reporting, to the number of fields you are reporting on (even if some are empty), and to the number of instruments. Specify only the date range, fields, and instruments that you need.

Fields with Static Values

Avoid requesting fields with static values (such as Asset Type and Instrument ID Type) in requests that return many rows of data, because those static fields will be unnecessarily retrieved each time. It is much better to request reference and other static data in a separate extraction request. Note that even empty fields require extraction time.

RIC Chains

Only request a RIC chain if you need most or all of the RICs in the chain. Tick History fully expands a chain to resolve it to all of its constituent RICS, so if you need only a few RICs, you can improve performance by requesting them individually.

CUSIP, ISIN, and SEDOLs

In many cases, one CUSIP, ISIN, or SEDOL maps to multiple RICs. The Time and Sales, Market Depth, and Intraday Summaries report types return data for all of a non-RIC identifier's corresponding RICs. If you do not want data for all of the RICs, map the non-RIC identifier to the set of RICs that you need, and report on only that set.

Note that by default, a request for Elektron Timeseries, Terms and Conditions, or Corporate Actions delivers data for the RIC corresponding to the specified CUSIP, ISIN, or SEDOL. If you also specify an exchange, it delivers data for the RIC corresponding to the specified CUSIP, ISIN, or SEDOL on that exchange. If you want results for several exchanges, you need one entry per exchange.

Querying Multiple Instruments Together Is Faster Than One at a Time

Submitting a single job that reports on multiple instruments is faster than submitting several jobs that each report on one instrument.

You can report on up to 30,000 RICs in a single report, and up to 5,000,000 RIC-days total in concurrent reports. For more information, see *Limits*.

Use a URL Based on Domain Name, Not on IP Address

When accessing Tick History, use a domain name URL (for example, https://hosted.datascope.reuters.com/DataScope/Home), not an IP address (which comprises only digits and periods). Using a URL based on domain name insulates you from changes to hosting infrastructure, and ensures that in the event of a failover to another hosting site you do not need to change scripts or browser bookmarks.

Get Output Files Faster by Downloading Them Directly From Amazon

- You can download files faster by retrieving them directly from the Amazon Web Services (AWS) cloud in which they are hosted.

  To do this via the GUI, select the user preference **Enable Direct Download From S3** in the **Tick History File Delivery** section of the **Preferences** screen.
- For information about doing this via the REST API, see the *Tick History REST API User Guide*.

Retrieve Reports When Completed, Not in the Sequence Submitted

Reports of the same type (for example, Tick History Market Depth reports) run sequentially, and complete in the order in which you submitted them. Reports of different types run in parallel (independently of each other) and complete when they are done.

Via the REST API, you can poll report jobs to determine when each one is done, which enables you to retrieve each one as soon as possible. Via the GUI, you can check the Completed Extractions tab to determine when each report is available.

Download Reports before They Become Unavailable

Extractions older than 45 calendar days are automatically deleted and cannot be recovered. Please download your Tick History extraction files upon receipt. **DataScope Select should not be as an archiving repository**. .Any file manipulation, such as changes to the file names and formats, should be performed post-download, on your own computer.

On-demand extractions expire after 7 days.

Use Extraction Notes to Troubleshoot Reports

All extraction requests return notes, such as information about quotas, embargoes, and permissions. These notes can be helpful to you when you troubleshoot problems. It’s a good idea to examine these notes on a regular basis and learn what kinds of information are available.

When using the REST API you can also search the notes for keywords that you see are associated with certain kinds of problems.

Avoid Open-Ended Searches

Tick History supports simple, ad-hoc searches and is not intended to be used as a discovery tool. Prefixed wildcards used in search queries will be ignored, as this practice is not an efficient use of the tool.

Java Support

The Tick History graphical user interface (GUI) and REST API support the same releases of Java Platform Standard Edition (usually referred to as Java SE) that are supported by Oracle's Premier Support program. You can see which Java SE releases are currently supported by Premier Support by going to the Oracle Java SE Support Roadmap table on the Oracle Java SE Support Roadmap page.

It is important that you use only supported versions of Java with DataScope Select. DataScope Select is tested and validated only with supported Java versions, and you may get non-standard behavior with other Java versions.

2 REST-API–Specific Best Practices

This chapter describes Tick History best practices that apply only to the REST API interface:

- Reuse Authentication Tokens for Faster Authentication
- Set a Realistic Polling Interval
- Do Not Resubmit After a Timeout
- Retrying REST API Error Responses
- Check Content-Encoding Header
- Avoid On-demand Extraction Requests for Repeated Tasks
- Make Non-Time-Critical Requests Outside of Market Close Times
- Disperse Requests Instead of Submitting Large Request Sets at Once
- Find Help and Support from the Refinitiv Developer Community

Reuse Authentication Tokens for Faster Authentication

Creating and reusing an authentication token is the fastest and most efficient way of handling authentication. It is superior to:

- Providing a user ID and password each time you submit a request.
- Creating a new authentication token each time you submit a request.

Set a Realistic Polling Interval

A large report that will take a long time to execute does not need a short polling interval. The more frequently you poll, the more system resources are consumed, so set the interval to an appropriate period relative to the report.

Do Not Resubmit After a Timeout

If your procedure that polls a report job times out on your system, do not resubmit the report job. It is only your polling procedure that has timed out: the original report job is still queued to execute, or is still executing, on the DataScope Select platform. Resubmitting the report job will not get you the report faster. (In fact, resubmitting it might have the opposite effect, because you will now have an additional instance of the job running.)

If your polling procedure frequently times out, consider increasing your timeout period.

Retrying REST API Error Responses

When making a request to the REST API, do not retry when the response has a status code of 501 or less. These errors are not recoverable.

For errors with a status code of 502 or greater, the retry logic should not retry more than three times per minute and no more than 10 times. Less is preferable so a downed server is not flooded with unserviceable requests.

Note that the C# Toolkit will automatically retry 502 or greater responses three times. It retries after one second. If still failing, the C# Toolkit will try again after five seconds. If still failing, the C# Toolkit will try one last time after 30 seconds. If this last call fails, an exception will be raised in the code.

Check Content-Encoding Header

File compression requests may be intermittently denied for various reasons with increased likelihood during periods of heavy use.

Use of the DataScope Select API SDK tool kit provides protection against compression request denials by system services. We recommend that clients not using the SDK tool kit check for the Content-Encoding parameter in the response header and do not attempt to unzip the results unless it is present.

Alternately, clients can use an unzip product that recognizes the compression status and only attempts to unzip if compressed. Always protect against compression request denials by one of these methods as a best practice, as compression requests cannot be 100% guaranteed.

Avoid On-demand Extraction Requests for Repeated Tasks

Scheduling determines when a report request will be processed (data extracted from the data sources). On-demand reports are acted upon immediately while stored and scheduled reports run by calendar day, week or day of the month at an hour of a day in a reoccurring manner.

Frequently repeated tasks, such as requesting the same data, at the same time, every day, should be performed using a stored and scheduled report and not an on-demand report. Using an on-demand report for this type of workflow may result in possible delays.

Make Non-Time-Critical Requests Outside of Market Close Times

Non-time-critical requests, including searches and extractions of terms and conditions and reference data, as well as historical pricing, should be done outside of regional market close times.

Disperse Requests Instead of Submitting Large Request Sets at Once

Disperse your requests instead of submitting a large set of requests at once. Submitting large sets of requests simultaneously can impede system performance. Please disperse your requests within the defined limits and only submit the requests that you need.

Find Help and Support from the Refinitiv Developer Community

The Refinitiv Developer Community provides Tick History API support and offers self-service learning through documentation and tutorials for performing authentication and setting up a development environment using Python and Postman.

The portal provides tools, documentation, sample code, learning materials and community Q & A forums to help you work effectively to build new, extended or integrated products using our open and controlled APIs.

3 Limits

Some Tick History limits apply universally to both the GUI and REST API interfaces, and some apply only to the REST API.

Limits Common to All Interfaces (REST API & GUI)

This section describes Tick History limits that do not distinguish between requests submitted via the GUI and via the REST API. When Tick History evaluates one of these limits, it sums the requests submitted from both of these interfaces.

- Prioritizing Extractions to Ensure Resource Availability for All Users
- Extraction Limits: Concurrent Requests per Report Template Type
- Extraction Limits: RIC-days in Concurrent Requests
- Extraction File Download Limits

Prioritizing Extractions to Ensure Resource Availability for All Users

Tick History ensures that one user who submits a large number of requests all at once against a given report template type cannot monopolize that template's resources at the expense of all other Tick History users. If other users submit requests against that template shortly after the first user, a queuing fairness algorithm ensures that the template executes requests for a mix of users, but still weights the mix of jobs to reflect the earlier submission time of the first user's requests.

Extraction Limits: Concurrent Requests per Report Template Type

Each API request triggers a function to run and return an acknowledgement response. While some functions, like reference requests, respond quickly, most create a job in a queue which subsequently runs at a time after the function call has received its HTTP response acknowledgement.

In order to support consistent performance and optimize response times for the most users, Tick History applies execution limits and queuing on a per-report template basis. You can submit a maximum of 50 concurrent requests per report template. The number of concurrent extractions that can be processed on a per-report and per-user basis is two for all report templates except Historical Reference and Elektron Timeseries report templates. Concurrent extraction processing is not supported for Historical Reference and Elektron Timeseries report templates.

Users are allowed to have up to four concurrent connections for downloads. Each connection may see speeds up to 1 MB/s.

Extraction Limits: RICs per Report Template Type

The maximum number of instruments that you can extract in a single request is determined by the report template type that you are using for the extraction. This limit is the same regardless of whether you are extracting the instruments using the REST API or the GUI.

Note that your input list of instruments can include different kinds of instrument identifiers (such as simple RICs, RIC chains, CUSIPs, ISINs, and SEDOLs). Tick History then expands and resolves this input list into a list of simple RICs:

- A RIC chain is expanded into its constituent simple RICs.
- Non-RIC instrument identifiers (such as CUSIPs, ISINs, or SEDOLs) are resolved to simple RICs. In some cases the mapping between other instrument codes and RICs is not 1:1.

The fully expanded and resolved list of RICs may contain more instrument identifiers than the original input list from which is was derived. For some report types, you are allowed more instrument identifiers in the fully resolved list than in the input list. For other report types, the limits on the fully resolved list and on the input list are the same.

For more information about the limits for the input list and for the fully resolved list, for each report type, please see *Extraction Limits* in the SDK menu on the Rest API Help site.

Note that this limit can be affected by the limit on the number of RIC-days in concurrent requests.

Extraction Limits: RIC-days in Concurrent Requests

Each user can extract up to 5,000,000 RIC-days in requests that are running at any one time.

For each user, Tick History maintains a tally of the total RIC-days being extracted by the user’s currently processing requests.

- If a user submits a request that increases his or her tally beyond that limit, Tick History rejects that new request.
- When one of the user’s currently-running requests ends (because it completes, fails, or is canceled), the request’s RIC-days are subtracted from the user’s “RIC-days” tally. Those reductions free up resources for the user to submit additional requests.
- The limit is evaluated separately for each Tick History user, that is, for each Tick History login account. The number of requests that one user is running does not affect the tallies of other users.

What is a RIC-day?

A “RIC-day” is a unit of measure in which data is extracted for one day for one RIC.

- A request’s total number of days is the number of days between the query start date and query end date, inclusive.
- A request’s total number of RICs is the sum of all fully-resolved RICs being extracted.
  - A RIC chain is counted as the number of RICs in the fully-resolved chain.
  - If the request uses instrument types other than RIC (such as SEDOL or CUSIP), the instruments of those other types are resolved to RICs, and it is the RICs that are counted against the limit. (In some cases the mapping between RICs and other instrument codes is not 1:1.)
- A request’s total number of RIC-days is its total number of days multiplied by its total number of RICs.

  For example, if a request reports on the period from July 8 to July 23, it covers 16 days. If it reports on three RICs, then 3 RICs * 16 days = 48 RIC-days.

To which report types does this limit apply?

This limit applies only to RIC-days in requests against these report types:

- Tick History Time and Sales
- Tick History Market Depth
- Tick History Intraday Summaries
- Tick History Raw

To which interfaces does this limit apply?

The limit does not distinguish between requests submitted via the GUI and via the API. The RIC-days in requests that are submitted via each interface are summed together.

When do a request’s RIC-days begin counting against my limit?

The precise time may vary by a minute or two, but for practical purposes:

- Assume that when you submit a request on-demand or schedule it to run immediately, its RIC-days will count against your limit as soon as the request is submitted.
- Assume that when a request is scheduled to run at a specified time, its RIC-days will count against your limit at that time.

When are the number of RIC-days evaluated?

When you submit a request, Tick History evaluates how many RIC-days the request would extract if it ran, and decides whether the request would cause the user to exceed his or her RIC-day limit given his or her RIC-day tally at the time of submission. If Tick History adds the submitted request’s RIC-days to the user’s current RIC-day tally and finds that it would increase the tally beyond the limit of 5,000,000 RIC days, Tick History rejects the submitted request before it has a chance to run.

Extraction File Download Limits

You can have up to four extraction downloads in progress at one time. This total applies to any combination of custom reporting and Venue by Day downloads, via any combination of the REST API and via the GUI.

There is an exception to extraction file download limits: there is no limit on the number of files that you can download *directly* from Amazon Web Services (AWS) at one time. Downloading directly from Amazon is described in the *Tick History REST API Users Guide* and the *Tick History User Guide*.

Limits Specific to the REST API

Authentication Token Limits

An authentication token remains valid for 24 hours. If a token has expired, you simply create a new one. You can reuse a token as often as you like within that 24-hour period. You can create as many tokens as you want, as often as you want.

Endpoint Frequency Limits

The Tick History REST API employs endpoint-based rate management to ensure consistent performance for all users.

Tick History measures the frequency of each REST API user’s requests against each endpoint. If your requests against a particular endpoint over a defined period of time (usually several seconds or minutes) exceed the per-user limit for that endpoint, each of your requests that is over the limit within that period of time will return HTTP status code 429 (Too Many Requests) with a description of the problem, and fail. You should wait several seconds and then resubmit the request, and should reduce the frequency with which you submit those requests.

Most users will rarely (if ever) reach these limits, because the limit for each endpoint represents a very heavy level of use.

Each frequency limit depends primarily on the number of requests that you submit against that endpoint within the defined period of time. But how Tick History applies that limit will depend also on secondary request processing factors that may be specific to your situation. So the limits shown below are approximate, because they are influenced by secondary factors. They are an absolute upper limit, but may vary by a small amount; we provide them to you to give you a sense of the range within which you will reach the limit. For example, a limit of 1,000 might indicate a range of approximately 850 to 1,000.

The table below shows frequency limits defined by category of endpoint with maximum request limits per time period. Both recommended and current maximum request limits are shown. (Recommended limits are provided for some endpoint categories; the others are forthcoming.) You are encouraged to code to the recommended maximum request limits. Doing so ensures optimal performance and reliability of the platform for all users.

Note that the limits below are current as of **13 July 2020.** They are likely to be reduced further.

Category	Recommended Maximum Requests	Current Maximum Requests	Time Period (seconds)	Endpoints
All Requests	500	2,000	60	All endpoints in the REST API.
Authentication		30	300	This comprises the endpoints in the Authentication folder in the REST API Reference tree.
Extractions, All	500	1,000	60	All extraction-related endpoints.
Extractions, On Demand	100	400	60	This comprises ExtractRaw, ExtractWithNotes, Extract, ExtractNow, and ExtractMultipleNow.
Extractions, Standard (includes VBD)		900	60	This comprises the endpoints in the StandardExtractions folder in the REST API Reference tree.
Miscellaneous (called **Category 6** in 429 error messages)		250	60	This comprises ExtractedFiles and GetPackageDeliveriesByDateRange.
On Demand Result Polling	30	200	60	All on-demand request polling, including ExtractResult, ExtractWithNotesResult, ExtractRawResult, and ExtractNowResult.
Users		400	300	This comprises the endpoints in the Users folder in the REST API Reference tree.
Search	30	1,500	60	This comprises the endpoints in the Users folder in the REST API Reference tree.

REFINITIV™