SPITZER API

WEBSOCKET SETUP GUIDE

Version 1.0 — June 2026

The Spitzer WebSocket API provides direct, bi-directional access to published news and live M&A data over a single multiplexed connection. The API supports real-time streaming (push) of news on publication and of live deal metrics on each market tick, plus request/response queries (pull) across both the news archive and the full quantitative dataset.

Technical support: Greg Falkowski — greg.falkowski@ctfn.news — +1 (609) 410-0556

1. CONNECTING TO THE API

Before connecting, ensure you have an active Spitzer account with API access enabled. Use the same email/username and password for the Spitzer API as you would for Spitzer Data or Spitzer News.

The WebSocket is authenticated via a bearer token, which you get first from the REST login endpoint.

Step 1 — Obtain a bearer token

POST your credentials to /login. The response returns an opaque bearer token that is valid for 24 hours; request a fresh token whenever a call returns 401.

```
POST https://spitzerdata.com/login
Content-Type: application/json

{ "username": "you@example.com", "password": "your_password" }

→  { "token": "<opaque-bearer>", "expires_in_hours": 24 }
```

Step 2 — Endpoint

Configure your client to open a WebSocket connection to:

```
wss://spitzerdata.com/ws
```

Authentication

Present the bearer token in EITHER the upgrade request's Authorization header or a ?token= query parameter (use the query parameter when your WebSocket client cannot set custom headers):

```
Authorization: Bearer <opaque-bearer>
—  or  —
wss://spitzerdata.com/ws?token=<opaque-bearer>
```

An invalid or missing token closes the socket with code 4401 (unauthorized). A valid token whose tier has no API entitlement is accepted, sent a no_access error, then closed with code 4403.

Welcome message

On a successful connection the server immediately sends a welcome frame describing your access level, whether you receive pushes, your pull budget, and the available channels:

```
{
  "type": "welcome",
  "level": "immediate",          // immediate | research_plus | research
  "can_push": true,              // true only at the Immediate level
  "pull_limit": null,            // max pulls per window (null = uncapped)
  "pull_window_seconds": null,
  "channels": ["news", "data"]
}
```

Access levels

Your access level is derived from your subscription tier and governs whether you receive pushes and how many pulls you may make. Pulls are charged against ONE per-user budget shared across every Spitzer API surface (WebSocket, MCP, REST and the =SPITZER Excel add-in).

Level	Push	Pull budget
`Immediate`	Yes — news + data	Uncapped
`Research+`	No	10,000 calls / 30 min
`Research`	No	50 calls / 10 min
`(none)`	No	No API access — connection rejected

Heartbeat

To check that a connection is alive, send a ping; the server replies with a keep-alive frame. (Unlike a server-initiated heartbeat, Spitzer's keep-alive is sent in response to your ping, so you control the cadence.)

```
→  { "type": "ping" }
←  { "type": "keep-alive" }
```

Reconnection

Client applications must reconnect automatically whenever they receive a WebSocket CLOSE event. Use exponential backoff (e.g. 1s, 2s, 4s, capped) to avoid thundering-herd reconnects, and re-send any data subscriptions after reconnecting (subscriptions are per-connection and are not persisted).

Example: connecting via wscat

```
wscat -c wss://spitzerdata.com/ws \
  --header Authorization:"Bearer YOUR_TOKEN"

  Connected (press CTRL+C to quit)
  < {"type":"welcome","level":"immediate","can_push":true,...}
  >
```

2. REAL-TIME STREAM API (PUSH)

Push delivery is available ONLY at the Immediate level. The server pushes a frame on the news channel whenever a story is published, and on the data channel whenever a subscribed ticker's

metrics change on the live tick. Observe the WebSocket MESSAGE event and branch on the top-level type / channel.

News push

Sent automatically to every Immediate connection the moment a Spitzer news story is published. No subscription is required for news.

```
{ "type": "push", "channel": "news", "content": { ... } }
```

Key	Type	Description
type	string	Always "push"
channel	string	Always "news"
content.title	string	Headline of the news story
content.author	string	Full name of the author
content.url	string	spitzernews.com URL for the story
content.slug	string	Stable slug (use to pull the full article)
content.tickers	array	Ticker symbols mentioned in the story
content.tags	array	Topic tags (e.g. Antitrust, CFIUS)
content.published	string	ISO-8601 publication timestamp

Data push

Sent on the live-price tick (~60s), but ONLY for tickers you have subscribed to and ONLY when a value has changed since the last push (the diff keeps quiet ticks silent). Each frame carries the full formatted parameter map for one ticker — the same shape as a data/all pull.

```
{ "type": "push", "channel": "data", "ticker": "AAPL",
  "content": { "break": "...", "spread": "...", "odds": "...", ... } }
```

Key	Type	Description
type	string	Always "push"
channel	string	Always "data"
ticker	string	The subscribed ticker this frame is for
content	object	Full {param: formatted-value} map for the ticker

Subscribing to data tickers

Data pushes need a subscription. Send a subscribe frame with one or more tickers; the server echoes back your full current subscription set. Send unsubscribe with tickers to remove them, or with no tickers to clear all. (Subscriptions are rejected with push_not_allowed below the Immediate level.)

```
→  { "type": "subscribe", "tickers": ["AAPL", "MSFT"] }
←  { "type": "subscribed", "channel": "data", "tickers": ["AAPL","MSFT"] }

→  { "type": "unsubscribe", "tickers": ["MSFT"] }
→  { "type": "unsubscribe" }            // clears ALL subscriptions
```

3. QUERYING THE API (PULL)

All levels (Immediate, Research+, Research) may pull. Send a JSON object with a query whose channel is either news or data. You may include an optional uid for your own request tracking; it is echoed back in the response. Each pull is charged one call against your budget.

Request format

```
{
  "uid": "optional-unique-id",
  "query": {
    "channel": "data",          // "data" | "news"
    "ticker":  "AAPL",          // data: a ticker;  news: filter by ticker
    "param":   "spread",        // data: a single parameter (omit for all)
    "all":     false,           // data: true → full parameter map
    "q":       "",              // data/news: search text
    "slug":    "",              // news: fetch one article by slug
    "limit":   25,              // news/search: max items
    "format":  "display"        // data: "display" | "raw"
  }
}
```

Query parameters

Parameter	Type	Description
`uid`	string?	Optional id, echoed back in the response
`query.channel`	string	Required. "data" or "news"
`query.ticker`	string?	data: the ticker to read; news: ticker filter
`query.param`	string?	data: a single parameter name (omit → all)
`query.all`	bool?	data: true returns the full parameter map
`query.q`	string?	data: search across params; news: full-text search
`query.slug`	string?	news: return a single article by slug
`query.limit`	int?	news / search: maximum items (default 25 / 20)
`query.format`	string?	data: "display" (default) or "raw"

4. RESPONSE FORMAT

A successful pull returns a result frame; a failed or rate-limited one returns an error frame. The result echoes your uid and channel, reports your window's remaining pull budget, and carries the value.

```
{
  "type": "result",
  "uid": "your-uid-if-provided",
  "channel": "data",
  "remaining": 9999,            // pulls left in the window (null = uncapped)
```

```
  "value": { ... }              // the requested data / news payload
}
```

Field	Type	Description
type	string	"result" on success, "error" on failure
uid	string?	Echoed from the request, if provided
channel	string	The channel queried ("data" / "news")
remaining	int?	Pulls left in the current window (null = uncapped)
value	any	The data value, value map, or news items
reason	string?	On error: rate_limited, push_not_allowed, pull_failed, …
retry_after	int?	On rate_limited: seconds until the window resets
limit	int?	On rate_limited: the window's call limit

Error example (rate limited)

```
{ "type": "error", "uid": "...", "reason": "rate_limited",
  "retry_after": 540, "limit": 50 }
```

5. CHANNELS

Data

The data channel exposes Spitzer's live M&A metrics — the same parameter set as the =SPITZER Excel function and the REST /api/ctfndata endpoints: break price, gross and all-in spreads, premium, implied gross / all-in chance of close, profit pool, closing guidance, comps and more. Read one parameter (param), the full map (all: true), or search across parameters (q). Values come formatted for display by default; pass format: "raw" for unformatted numerics.

News

The news channel returns published Spitzer / CTFN news stories covering M&A activity, regulatory developments and market-moving events. List and filter by ticker or full-text query (q), or fetch a single story's full text by slug.

6. MCP TOOL INTEGRATION

For AI-assisted workflows, Spitzer provides five MCP (Model Context Protocol) tools that wrap the same news and data surface into structured, callable functions. The MCP server manages the bearer token and re-authenticates automatically on expiry, so agents never handle the login flow directly.

Tool	Kind	Description
spitzer_data	data	One parameter for one ticker (ticker, param, format)

spitzer_data_all	data	The full parameter map for a ticker (ticker)
spitzer_data_search	data	Search parameters across deals (query, param, limit)
spitzer_news	news	Search / list news (query, ticker, limit)
spitzer_news_article	news	Fetch one article's full text (slug)

7. BEST PRACTICES

Pull budget

- One call = one logical request, counted identically on every surface (WebSocket, MCP, REST, Excel). The budget is per user and shared across all of them.
- Immediate is uncapped. Research+ allows 10,000 calls / 30 min; Research allows 50 / 10 min.
- Read remaining on each result to pace yourself; on a rate_limited error, wait retry_after seconds before retrying.

Streaming vs. pulling

- At the Immediate level, prefer pushes for anything you watch continuously — subscribe to data tickers and listen for news, rather than polling.
- Data pushes are diff-gated: you only receive a frame when a subscribed ticker's value changes.
- Below Immediate there are no pushes — pull on your own schedule, within budget.

Connection management

- Maintain a single persistent WebSocket connection rather than reconnecting per query.
- Reconnect automatically with exponential backoff on CLOSE, and re-send your data subscriptions afterwards (they are per-connection).
- Send a periodic ping and watch for the keep-alive reply to detect a stale connection.

8. QUICK REFERENCE

Item	Value
Endpoint	wss://spitzerdata.com/ws
Protocol	WebSocket (WSS)
Authentication	Bearer token — Authorization header or ?token=
Get a token	POST /login { username, password } → 24h bearer
Channels	news, data
Server frames	welcome, result, error, push, keep-alive, subscribed
Push (Immediate only)	news on publish; data on tick for subscribed tickers
Subscribe	{ "type":"subscribe", "tickers":[...] }
Pull	{ "uid":"…", "query":{ "channel":"data"|"news", … } }
Heartbeat	send { "type":"ping" } → { "type":"keep-alive" }
Close codes	4401 unauthorized · 4403 no API access for tier
Pull budgets	Immediate uncapped; Research+ 10k/30m; Research 50/10m