WET DOG
WEATHER

wetdogweather.com

PRODUCT

WEB COVERAGE SERVICE

What Is Web Coverage Service?

Web Coverage Service is an OGC standard for accessing gridded datasets. It's structured in a way that encourages the use of weather data. Lots of support for big data lists, along with ways to acknowledge that data may be missing sometimes.

Wet Dog Weather uses the base WCS standard along with the metocean metadata extensions.

Wet Dog Weather supports three basic types of queries:

1. What data sets are available (GetCapabilities)
2. Specific information about a data set (DescribeCoverage)
3. Individual or gridded values from a specific data set (GetCoverage)

To a certain extent, WCS is self-documenting. Or at least it's easier to understand from the returns than it is from the specification. We suggest just making up a few queries and looking at the returns. There are even shortcuts we'll discuss later.

Your WCS Endpoint & TL:DR

Our customers have their own endpoints for WCS, as they do for our other products. These will allow access to all of their data, everything they're using from our standard data sets, and anything they may be buying or providing.

We'll give you an endpoint for our services of the form *<company>*.api.wetdogweather.com. In the rest of this document, we'll refer to that URL as *{{service}}*.

Excited to get started? Try out these URLs with your *{{service}}* and see what you get from a model everyone is likely to have access to: GFS.

Query	**URL**
Capabilities	{{service}}/wcs/GetCapabilities
DescribeCoverage	{{service}}/wcs/DescribeCoverage?CoverageId=gfs-global-atmos_latest
GetCoverage	{{service}}/wcs/GetCoverage?SUBSET=lat(40)&SUBSET=long(-98)&FORMAT=JSON&CRS=WGS:84&RANGESUBSET=temperature&CoverageId=gfs-global-atmos_latest&SUBSET=z(2)
GetCoverage	{{service}}/wcs/GetCoverage?SUBSET=lat(40)&SUBSET=long(-98)&FORMAT=JSON&CRS=WGS:84&RANGESUBSET=cloud_cover&CoverageId=gfs-global-atmos_latest

That second-to-last one fetches the whole forecast run at the given location for 2m temperature. The last one fetches cloud cover for the whole GFS run. We'll explain those calls in more detail.

Authorization

We do support authorization tokens now, but we don't require them. We use the standard authorization header in an HTTP call, but we can also accept an authorization parameter (*&auth=<nonsense string>*) in the URL.

At some point, we will turn on authorization for WCS, so it's best to build it into your services now. We can give you auth tokens to use as you like.

Hiding Our Endpoints & Building Your Own Service

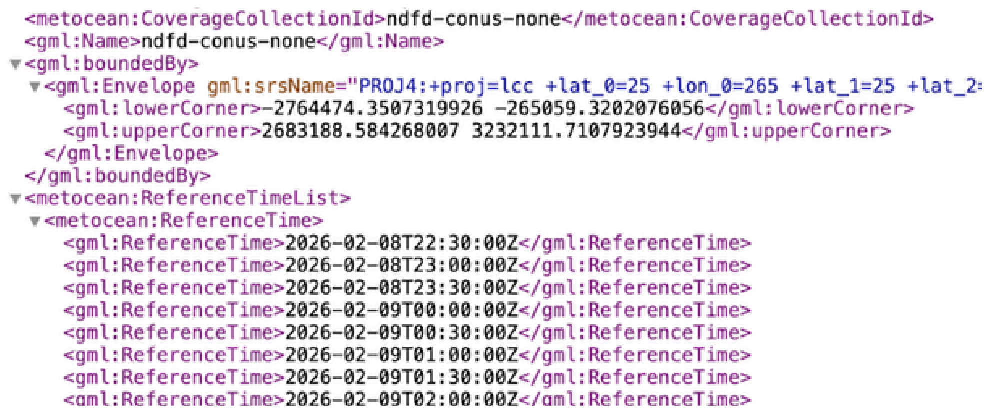
If you're white-labeling our service (we don't mind), you'll have a set of endpoints similar to ours. The simplest approach is just to forward the /wcs calls on to our service. The endpoint we give you is stable and includes a CDN to cut down on traffic.

You can build a real-time service on top of our endpoints. It's designed for that, but you'll probably want to stick with GetCoverage calls, as those are the fastest. Anything involving metadata, GetCoverage in particular, scales with the amount of data. If you're using radar, this can get large quickly.

GetCapabilities: What Data Sets are Available

Like many OGC standards, you can ask the server for a full list of what's available. WCS has better ways of doing this, which can reduce bloat. We're using the metocean extension, which is very compact and looks like this.

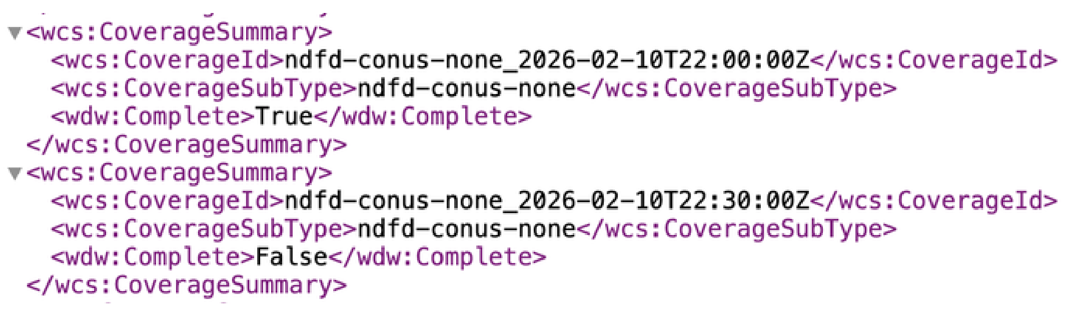
Here's the result of part of this query: {{service}}/wcs/GetCapabilities



```
  <metocean:CoverageCollectionId>ndfd-conus-none</metocean:CoverageCollectionId>
  <gml:Name>ndfd-conus-none</gml:Name>
▼<gml:boundedBy>
  ▼<gml:Envelope gml:srsName="PROJ4:+proj=lcc +lat_0=25 +lon_0=265 +lat_1=25 +lat_2
      <gml:lowerCorner>-2764474.3507319926 -265059.3202076056</gml:lowerCorner>
      <gml:upperCorner>2683188.584268007 3232111.7107923944</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
▼<metocean:ReferenceTimeList>
  ▼<metocean:ReferenceTime>
      <gml:ReferenceTime>2026-02-08T22:30:00Z</gml:ReferenceTime>
      <gml:ReferenceTime>2026-02-08T23:00:00Z</gml:ReferenceTime>
      <gml:ReferenceTime>2026-02-08T23:30:00Z</gml:ReferenceTime>
      <gml:ReferenceTime>2026-02-09T00:00:00Z</gml:ReferenceTime>
      <gml:ReferenceTime>2026-02-09T00:30:00Z</gml:ReferenceTime>
      <gml:ReferenceTime>2026-02-09T01:00:00Z</gml:ReferenceTime>
      <gml:ReferenceTime>2026-02-09T01:30:00Z</gml:ReferenceTime>
      <gml:ReferenceTime>2026-02-09T02:00:00Z</gml:ReferenceTime>
```

That's a concise way of saying we have NDFD data for this set of time steps. It doesn't list the variables we support, but there's another call for that.

We also support the older way of doing it, listing each model run individually.



```
▼<wcs:CoverageSummary>
    <wcs:CoverageId>ndfd-conus-none_2026-02-10T22:00:00Z</wcs:CoverageId>
    <wcs:CoverageSubType>ndfd-conus-none</wcs:CoverageSubType>
    <wdw:Complete>True</wdw:Complete>
  </wcs:CoverageSummary>
▼<wcs:CoverageSummary>
    <wcs:CoverageId>ndfd-conus-none_2026-02-10T22:30:00Z</wcs:CoverageId>
    <wcs:CoverageSubType>ndfd-conus-none</wcs:CoverageSubType>
    <wdw:Complete>False</wdw:Complete>
  </wcs:CoverageSummary>
```

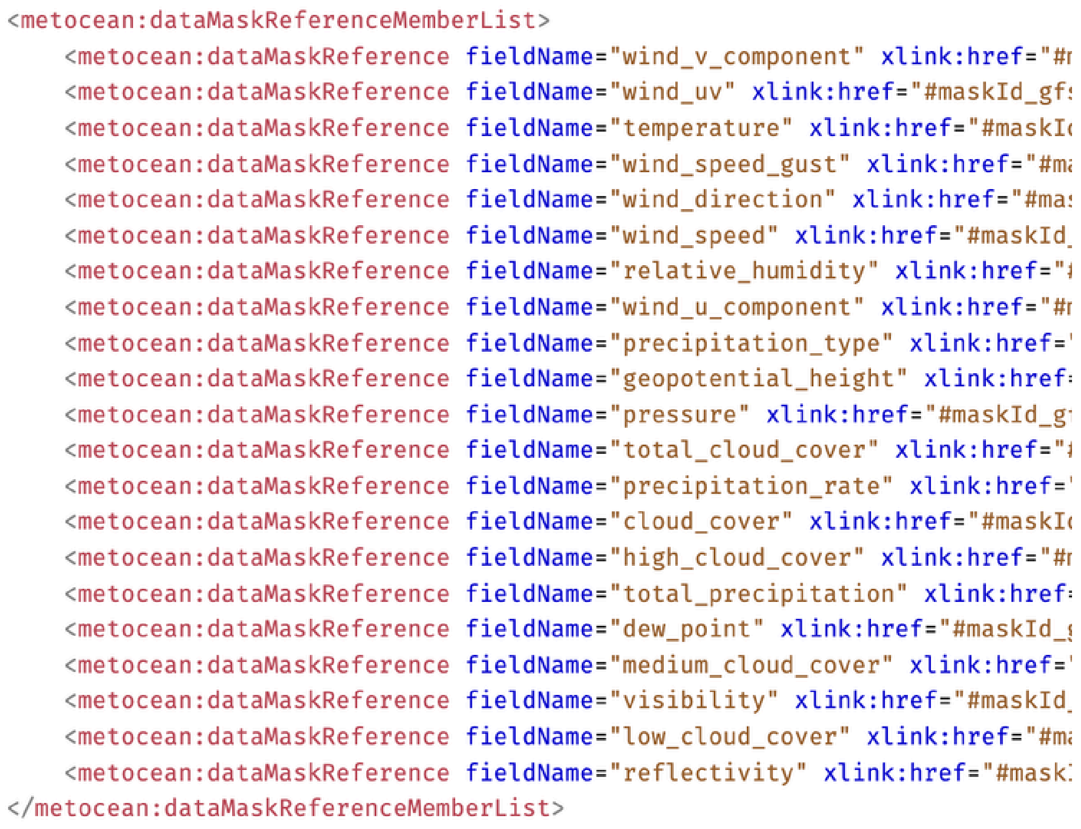
This is a bit wordier, but it does have the advantage of indicating which model runs have all data (*complete is True*) and which are still coming in (*complete is False*). That flag isn't in the spec; we added it for convenience.

The return XML can get a little large. We're seeing as much as 10MB for some of our users. Return times are around 5s for that use case, so plan accordingly. Our WCS API is meant for real-time use, so this can be a bit of a burden. We have ways of working around this problem, which we'll talk about later.

What's in a Data Set

DescribeCoverage is the call you can use to examine a model run more closely. If we pass the CoverageId *ndfd-conus-non_latest-complete* as above, we'll receive about 54 KB of XML. That's parseable and usable in real time if you're so inclined.

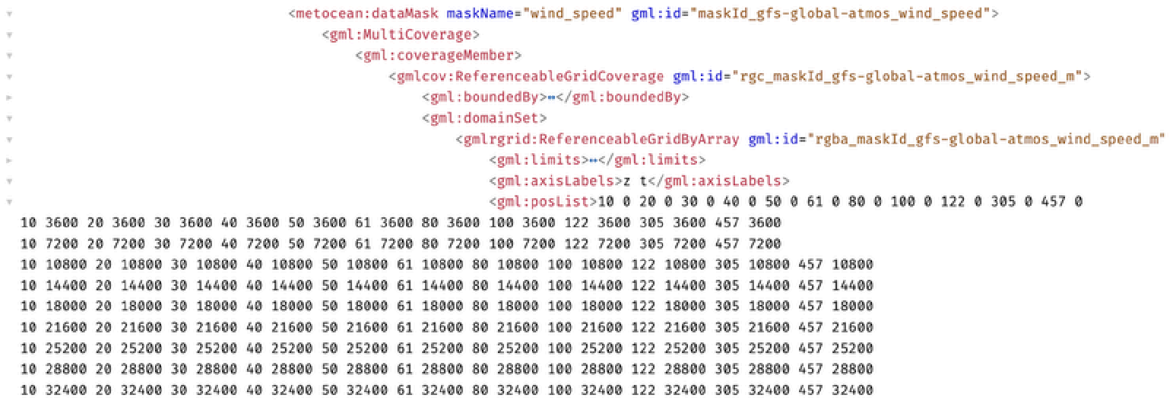
The DescribeCoverage return is... a bit complicated. Probably the most useful bit is the list of variables.



```
<metocean:dataMaskReferenceMemberList>
    <metocean:dataMaskReference fieldName="wind_v_component" xlink:href="#r
    <metocean:dataMaskReference fieldName="wind_uv" xlink:href="#maskId_gf
    <metocean:dataMaskReference fieldName="temperature" xlink:href="#maskI
    <metocean:dataMaskReference fieldName="wind_speed_gust" xlink:href="#ma
    <metocean:dataMaskReference fieldName="wind_direction" xlink:href="#ma
    <metocean:dataMaskReference fieldName="wind_speed" xlink:href="#maskId_
    <metocean:dataMaskReference fieldName="relative_humidity" xlink:href="
    <metocean:dataMaskReference fieldName="wind_u_component" xlink:href="#r
    <metocean:dataMaskReference fieldName="precipitation_type" xlink:href="
    <metocean:dataMaskReference fieldName="geopotential_height" xlink:href
    <metocean:dataMaskReference fieldName="pressure" xlink:href="#maskId_g
    <metocean:dataMaskReference fieldName="total_cloud_cover" xlink:href="
    <metocean:dataMaskReference fieldName="precipitation_rate" xlink:href="
    <metocean:dataMaskReference fieldName="cloud_cover" xlink:href="#maskI
    <metocean:dataMaskReference fieldName="high_cloud_cover" xlink:href="#r
    <metocean:dataMaskReference fieldName="total_precipitation" xlink:href
    <metocean:dataMaskReference fieldName="dew_point" xlink:href="#maskId_
    <metocean:dataMaskReference fieldName="medium_cloud_cover" xlink:href="
    <metocean:dataMaskReference fieldName="visibility" xlink:href="#maskId_
    <metocean:dataMaskReference fieldName="low_cloud_cover" xlink:href="#ma
    <metocean:dataMaskReference fieldName="reflectivity" xlink:href="#mask
</metocean:dataMaskReferenceMemberList>
```

This spec uses internal XML references, which were perhaps an unfortunate choice XML made long ago. Odds are you don't need to parse it, just look at it to decide what data is available and use that to construct the queries you'll code.

To that end, here's the *wind_speed* entry from the DescribeCoverage return. Keep in mind that DescribeCoverage covers a single forecast run (e.g., GFS 12z).



```
<metocean:dataMask maskName="wind_speed" gml:id="maskId_gfs-global-atmos_wind_speed">
    <gml:MultiCoverage>
        <gml:coverageMember>
            <gmlcov:ReferenceableGridCoverage gml:id="rgc_maskId_gfs-global-atmos_wind_speed_m">
                <gml:boundedBy>…</gml:boundedBy>
                <gml:domainSet>
                    <gmlrgrid:ReferenceableGridByArray gml:id="rgba_maskId_gfs-global-atmos_wind_speed_m"
                        <gml:limits>…</gml:limits>
                        <gml:axisLabels>z t</gml:axisLabels>
                        <gml:posList>10 0 20 0 30 0 40 0 50 0 61 0 80 0 100 0 122 0 305 0 457 0
10 3600 20 3600 30 3600 40 3600 50 3600 61 3600 80 3600 100 3600 122 3600 305 3600 457 3600
10 7200 20 7200 30 7200 40 7200 50 7200 61 7200 80 7200 100 7200 122 7200 305 7200 457 7200
10 10800 20 10800 30 10800 40 10800 50 10800 61 10800 80 10800 100 10800 122 10800 305 10800 457 10800
10 14400 20 14400 30 14400 40 14400 50 14400 61 14400 80 14400 100 14400 122 14400 305 14400 457 14400
10 18000 20 18000 30 18000 40 18000 50 18000 61 18000 80 18000 100 18000 122 18000 305 18000 457 18000
10 21600 20 21600 30 21600 40 21600 50 21600 61 21600 80 21600 100 21600 122 21600 305 21600 457 21600
10 25200 20 25200 30 25200 40 25200 50 25200 61 25200 80 25200 100 25200 122 25200 305 25200 457 25200
10 28800 20 28800 30 28800 40 28800 50 28800 61 28800 80 28800 100 28800 122 28800 305 28800 457 28800
10 32400 20 32400 30 32400 40 32400 50 32400 61 32400 80 32400 100 32400 122 32400 305 32400 457 32400
```

You can reliably parse this if you need to, but odds are you just want a little information out of it. This entry (*axisLabels*) lists the heights and times available (*posList*) in a series of numbers. First is the height (10m, for instance), followed by the time in seconds for the forecasts.

Models often have different forecast lengths, but those are known ahead of time. You can verify that information from the forecast offsets and get the list of heights we have available for a given variable.

That tends to be enough to build your queries, and WCS can deal with a bit of ambiguity. In fact, we've built some in for your convenience.

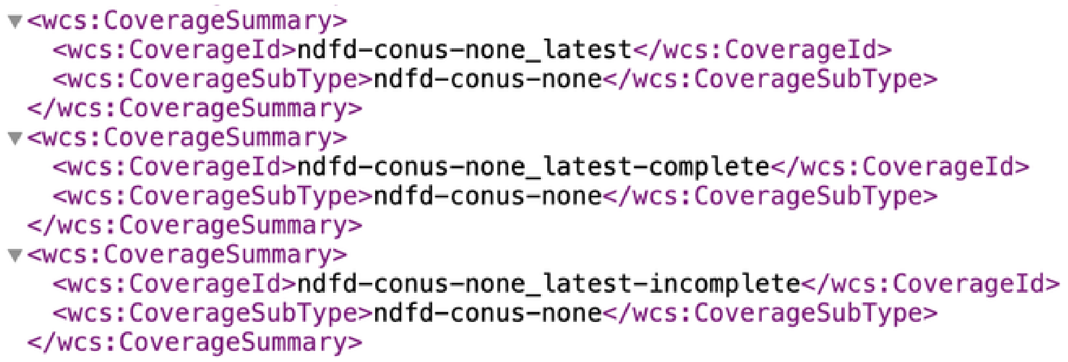
Latest vs Incomplete

With many of these OGC standards, the Capabilities function is well-intentioned but slower than you'd like. So users and providers develop shortcuts. Ours is the "latest" family of queries.

Keyword	**Purpose**
latest-complete	Returns information about the latest completed model run (e.g., GFS 12z)
latest-incomplete	Returns whatever it has on the most recent model run. Might be completed, might not. If incomplete, you won't get a full forecast run back.
latest	Currently mapped to latest-complete, but we plan to change this to a combination of latest-complete and latest-incomplete.

This is one of the areas where WCS begins to behave less like a legacy standard and more like a real-time weather data API. You're not waiting for full file transfers. You're querying what exists right now.

Normally, a model run has the - separated name with a time tacked on the end. In the case of the latest keywords, those are used rather than the time. We also list these options in Capabilities.



```
▼<wcs:CoverageSummary>
   <wcs:CoverageId>ndfd-conus-none_latest</wcs:CoverageId>
   <wcs:CoverageSubType>ndfd-conus-none</wcs:CoverageSubType>
 </wcs:CoverageSummary>
▼<wcs:CoverageSummary>
   <wcs:CoverageId>ndfd-conus-none_latest-complete</wcs:CoverageId>
   <wcs:CoverageSubType>ndfd-conus-none</wcs:CoverageSubType>
 </wcs:CoverageSummary>
▼<wcs:CoverageSummary>
   <wcs:CoverageId>ndfd-conus-none_latest-incomplete</wcs:CoverageId>
   <wcs:CoverageSubType>ndfd-conus-none</wcs:CoverageSubType>
 </wcs:CoverageSummary>
```

That's a lot of discussion of metadata, and most users will just look at it occasionally and hard-code the queries they need. Here's how to do that.

GetCoverage Call

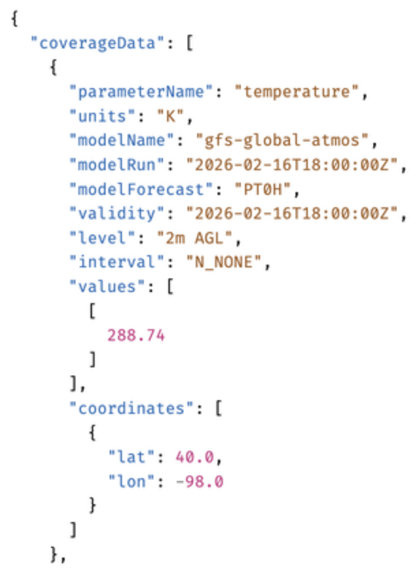
The workhorse of WCS is the GetCoverage call. Two types are available, one to get values at a specified coordinate (a point) and one to get a NETCDF image for a given box covering the data. The difference comes from the specified SUBSET coordinate values. The box option is limited to one time slice. We'll get to the details, but let's just look at a few examples.

Purpose	**URL**
Temperature 2m All Hours	{{service}}/wcs/GetCoverage?SUBSET=lat(40)&SUBSET=long(-98)&FORMAT=JSON&CRS=WGS:84&RANGESUBSET=temperature&CoverageId=gfs-global-atmos_latest&SUBSET=z(2)
Wind Speed Wind Direction 10m All Hours	{{service}}/wcs/GetCoverage?SUBSET=lat(40)&SUBSET=long(-98)&FORMAT=JSON&CRS=WGS:84&RANGESUBSET=wind_speed,wind_direction&CoverageId=gfs-global-atmos_latest&SUBSET=z(10)
Visibility All Hours	{{service}}/wcs/GetCoverage?SUBSET=lat(40)&SUBSET=long(-98)&FORMAT=JSON&CRS=WGS:84&RANGESUBSET=visibility&CoverageId=gfs-global-atmos_latest
Wind Gust First 6 Hours	{{service}}/wcs/GetCoverage?CoverageId=gfs-global-atmos_latest&SUBSET=lat(37)&SUBSET=long(-76)&FORMAT=JSON&RANGESUBSET=wind_speed_gust&CRS=WGS:84&SUBSET=f(PT0H,PT5H)

Pressure All Hours	{{service}}/wcs/GetCoverage? SUBSET=lat(40)&SUBSET=long(-98)&FORMAT=JSON&CRS=WGS:84&RANGESUBSET=pressure&CoverageId=gfs-global-atmos_latest
Temperature Bounding Box Single Hour	{{service}}/wcs/GetCoverage? FORMAT=NETCDF&RANGESUBSET=temperature&CRS=WGS:84&CoverageId=gfs-global-atmos_latest&SUBSET=z(80)&SUBSET=f(PT1H)&SUBSET=long(-86,-83)&SUBSET=lat(40,43)

Each of these uses WCS slightly differently, but they're all pointed at *gfs-global-atmos_latest*. For documentation purposes, that's best. We know that coverageId will work. But you can identify any valid coverage like: *gfs-global-atmos_2026-02-10T18:00:00Z*

Picking the temperature first example above, the first entry in the return would look like this.



```
{
  "coverageData": [
    {
      "parameterName": "temperature",
      "units": "K",
      "modelName": "gfs-global-atmos",
      "modelRun": "2026-02-16T18:00:00Z",
      "modelForecast": "PT0H",
      "validity": "2026-02-16T18:00:00Z",
      "level": "2m AGL",
      "interval": "N_NONE",
      "values": [
        [
          288.74
        ]
      ],
      "coordinates": [
        {
          "lat": 40.0,
          "lon": -98.0
        }
      ]
    },
```

It's JSON! Not surprising since we asked for JSON, but much friendlier to parse. There are other options, like NetCDF.

What's often as interesting with WCS is what you don't include. We didn't include a forecast time in the query, so we got all the forecast times back from the latest model run.

This is where WCS fully acts as a real-time weather data API. You're not downloading full model grids unless you want to. You're querying exactly what you need, at a specific point, for a specific time range, in a format your application can use immediately.

Picking the last example above, we have a NETCDF query, which will return a NETCDF image for the given time slice, level, and specified box.

GetCoverage Parameters

This call is where things get interesting, and there are a lot of options. We follow the specification for most of them, but here are the ones we implement.

Parameter	**Meaning**
CoverageId	The unique identifier for the model or data set to query. This might be something like *gfs-global-atmos_2026-02-10T18:00:00Z* or just *gfs-global-atmos_latest* as discussed earlier.
SUBSET	Used to define elevation or temporal parameters. See the next table.
FORMAT	*JSON* is the most common and the one we've built the speed around. *netcdf* is also available for bounding box queries.
RANGESUBSET	The variable names. One or more variables listed, separated by commas. This would be something like *temperature* or *wind speed*.
CRS	The coordinate system location is specified in. This will generally be *WGS:84* if you're specifying latitude and longitude.

Within the SUBSET keyword, there are quite a few options. They generally limit data by location, by height, or by time. They are always of the form: *param(value)*.
For example: *lat(39)*.

SUBSET	**Purpose**
lat, long	Latitude and longitude values. These are assumed to be degrees unless you set the CRS to something else. In the URL, they'll be of the form: *SUBSET=lat(40)&SUBSET=long(-98)* Either lat, long, or x, y is required. For a point query, it will include a single value like lat(45). For a box query, it will include two values: the first one for the lower left corner and the second one for the upper right corner of the box. For example, lat(40,43), 40 is the latitude of the lower left corner of the box, and 43 is the latitude of the upper right corner
x, y	If you're not using degrees, you can use the local coordinate system of the data if you don't specify a CRS. Or you can specify in X and Y of the CRS you pass in. Box query works the same way as (lat, long) query for box
f	The forecast hour(s) to fetch. If left off, the service will return all forecast hours for the specified run. Syntax for a single hour is like this: *f(PT10H)* For a range it's like this: *f(PT0H,PT6H)*

	Forecast time can be specified as Hour, Minute, Second, or any combination of those. Here is the 2hr 3min forecast offset: *f(PT2H30M)*
t	This is absolute time, rather than a forecast offset. Use UTC times in the same form we've seen before. For example, to get PT1H of the GFS we've seen before: *t(2026-02-10T19:00:00Z)*
z	Height in meters. This is the most common way to fetch a particular level. If we wanted 2m temperature, we would use: *z(2)*
is	Isobaric (pressure) levels. For data that comes in isobaric levels, you can retrieve them individually with this keyword, like so: *is(2)*
fl	Flight levels. For data that comes in flight levels, you can retrieve them like so: *fl(10)*
hy	These are the HRRR hybrid isobaric levels used in that model. If you want to retrieve them like so: *hy(2)*
msl	Height is the mean sea level. Doesn't require a value.
tro	Height is the tropopause. Doesn't require a value.

At the very least, you must specify some sort of location information. You can leave height and forecast time empty, in which case you'll get all of whichever you didn't specify. There are more heights in models than you'll typically use, so it's good to at least specify height.