

Wolfram|Alpha API Documentation

Preliminary, Rev. 0.7
June 2, 2009

The Wolfram|Alpha service provides a web-based API for clients to integrate the computational and presentation capabilities of Wolfram|Alpha into their own applications or web sites.

The API provides two general classes of queries. At the highest level, you can submit free-form queries like users might enter at the Wolfram|Alpha site itself, and get back full Wolfram|Alpha output in a variety of formats. The second type of query is a lower-level request for a single well-defined result, or range of results, from our entity/property-based data API, such as a caloric value for a food item, or a tide table for a requested location. These two API levels are quite different and are treated in separate sections below.

Both levels of the API are implemented in a standard REST protocol using HTTP GET requests. The result comes back as a descriptive XML structure wrapping the requested content format.

This is a preliminary document, and some details of the API may change before the API becomes available. New details and features will certainly be added, along with more extensive documentation and sample code. This document also does not deal with licensing considerations. Certain parts of the API or capabilities might not be available to all clients depending on licensing issues.

Obtaining an AppID

Visit the Wolfram|Alpha developer site (<http://www.wolframalpha.com/apiapplication.html>) to register for the Developer Program and obtain an appid. An appid is a string that identifies your application or organization, and it must be supplied in all calls to the Wolfram|Alpha API.

The “Query” API

The highest-level API is called the Query API because it allows callers to supply free-form natural language queries identical to what you would type into the Wolfram|Alpha web site itself. This high-level API lets callers retrieve full Wolfram|Alpha output in a variety of formats. The standard format is text and images, but you can also get HTML with CSS and JavaScript if you want the same formatting and behavior as on the Wolfram|Alpha site itself. This makes it very easy for clients to embed formatted Wolfram|Alpha output directly into their own web pages.

There are two functions in the Query API: `query` and `validatequery`. The `query` function is the main function, and it is accessed by a URL that begins with `http://api.wolframalpha.com/v1/query`. Its input and output are described in detail in the next few sections. In contrast, the `validatequery` function is a specialized function that performs only the initial parsing phase of Wolfram|Alpha processing. It can be used to quickly determine whether Wolfram|Alpha can make sense of a given input, bypassing the more time-consuming stages of fully analyzing the input and preparing results. The `validatequery` function is described in a later section.

Basics of Wolfram|Alpha Output

To understand the workings of the API, it is necessary to know some details about Wolfram|Alpha output. The screen shot below is a very simple example showing partial output for a user's entry of pi, as seen on the Wolfram|Alpha site. The output is divided into rectangular regions called pods, each of which corresponds roughly to one category of result. The output pictured below has five pods. Each pod has a title ("Input" is the title of the first pod), and content, which is typically a GIF image by default. Pods may also have additional features such as a copyable plaintext representation that appears in a popup when you mouse over the image, and JavaScript buttons that replace the pod with different information in an AJAX-style operation (like the "More digits" button in the output below).

The box at the top that starts with "Assuming pi is a mathematical constant" is not a pod, but rather a means to interact with Wolfram|Alpha's Assumptions facility. Assumptions are described in more detail later.

The screenshot shows the WolframAlpha interface for the query "pi". At the top is the WolframAlpha logo with the tagline "computational... knowledge engine". Below the logo is a search bar containing the text "pi". Underneath the search bar is a box with the text "Assuming 'pi' is a mathematical constant | Use as a character or a movie or a word instead". The main output area consists of five pods:

- Input:** π (with a "Mathematica form" link)
- Decimal approximation:** 3.141592653589793238462643383279502884197169399375105820... (with a "More digits" link)
- Property:** π is a transcendental number
- Continued fraction:** [3; 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 1, 84, 2, 1, ...] (with "Fraction form" and "More terms" links)
- Alternative representation:**
 - $\pi = 180^\circ$
 - $\pi = -i \log(-1)$
 - $\pi = 2 E(0)$

$E(x)$ is the complete elliptic integral of the second kind »
 i is the imaginary unit »
 $\log(x)$ is the natural logarithm »
[More information »](#)

Series representation: [More](#)

$$\pi = 4 \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

$$\pi = -2 + 2 \sum_{k=1}^{\infty} \frac{2^k}{\binom{2k}{k}}$$

$$\pi = \sum_{k=0}^{\infty} \frac{50k-6}{2^k \binom{3k}{k}}$$

$\binom{n}{m}$ is the binomial coefficient »
[More information »](#)

Pods have subpods that enclose the actual content. The pod titled "Alternative representation" in the screen shot has three subpods, each one showing a different mathematical identity for π . Each subpod is a separate result and a separate image on the page. By convention, every pod has at least one subpod, so pods that appear to show only one result have that result in a subpod.

On the Wolfram|Alpha site, the content of each subpod is generally an image (even the π in the Input pod above is a GIF image, not text). Most results have alternative formats, such as various forms of textual representation. Users of the API can request any combination of these different types of representations.

Output Formats

The Wolfram|Alpha API provides a number of formats in which results can be returned. You can request any combination of formats. The result of a call to the API is always an XML document, with each pod and/or subpod represented in one or more of the following formats. More details on the format of the result XML are provided later; this is just a summary of the available format types.

Visual Representations

On the Wolfram|Alpha interactive web site, results are displayed in the form of GIF images. This allows mathematical formulas, tables, and of course graphics, to be formatted in a meaningful and attractive way. When using the Wolfram|Alpha API, you have two choices if you want such "pictures" of the output.

Image

The "image" format gives you the same types of GIF images as seen on the Wolfram|Alpha site. Each subpod is returned as an HTML `` tag ready for direct inclusion in a web page.

HTML

Each pod is returned in the form of HTML, just as it is on the Wolfram|Alpha site itself. You also get HTML for the CSS and JavaScript inclusions necessary to make the output appear and behave like it does on the Wolfram|Alpha site.

PDF

Each subpod is returned as a URL to a PDF file.

Textual Representations

In some cases, you might not be satisfied with pictures of output, but will instead want some sort of structured textual representation of each subpod. You can then format it in a custom way, or pick it apart to extract only the desired piece. Not all results are available in all the formats listed below. For example, a plot of a mathematical function will have no plaintext representation, although it will have a *Mathematica* Input representation.

Plaintext

This is the text format that you see in the "Copyable plaintext" popup that appears when you click on results on the Wolfram|Alpha site. It represents a simple "best guess" for a meaningful readable textual form of a given result. You will find it difficult, if not impossible, to write general-purpose code to analyze text in this format, so use it only if you want simple text to display to your users, or if you know the structure of the text in advance (e.g., it will be a number, or a latitude-longitude pair, etc.)

Mathematica Input

This is the text format that you see in the "*Mathematica* plaintext input" popup that appears when you click on some results on the Wolfram|Alpha site. *Mathematica* is the computational engine that powers Wolfram|Alpha, and many results can be generated directly by single *Mathematica* input expressions. For example, the "Continued fraction" pod in the earlier Wolfram|Alpha screen shot for pi has a *Mathematica* input representation of `ContinuedFraction[Pi, 25]`. Use this form if you want to feed the input into *Mathematica*, or use *Mathematica* as an environment for manipulating results.

Mathematica Output

This is the text format that you see in the "*Mathematica* plaintext output" popup that appears when you click on some results on the Wolfram|Alpha site. Because all Wolfram|Alpha results exist in the form of *Mathematica* expressions before they are formatted, this format is available for all results, although it will sometimes be large (e.g., for mathematical plots), or not very useful (e.g., when the original source data is only available to *Mathematica* as a raster image, such as a country's flag). The first formula in the "Continued fraction" pod in the earlier Wolfram|Alpha screen shot for pi has a *Mathematica* output representation of `{3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 2, 1, 84, 2, 1, 1}`. Use this form if you want to feed the output into *Mathematica*, or use *Mathematica* as an environment for manipulating results.

MathML

Some Wolfram|Alpha results are mathematical expressions or formulas that require traditional math notation to look good (superscripts, fractions, integral signs, etc.) Presentation MathML (<http://www.w3.org/Math>) is a W3C standard XML format for mathematics. Many browsers can render MathML, perhaps with the assistance of a plug-in.

ExpressionML

ExpressionML (<http://reference.wolfram.com/mathematica/ref/format/ExpressionML.html>) provides a means to represent an arbitrary *Mathematica* expression in XML. There is a one-to-one correspondence between the ExpressionML representation and the *Mathematica* Output representation described above. Choose this format if you need to analyze or decompose the structure of results, and you prefer to use XML-based tools or libraries.

XML

Although ExpressionML is a structured XML representation of a result, it precisely represents the *Mathematica* output expression, and is more suitable for programmers who are familiar with *Mathematica* expression structure. For some types of results, such as tables, a more natural XML representation is also available. More details on this XML format are forthcoming.

Sample Query

A simple API call to retrieve output for the query "pi" would look like this:

```
http://api.wolframalpha.com/v1/query?input=pi&appid=xxxxx
```

This query did not specify a desired output format, and the default is to retrieve the plaintext and image representations of each subpod. The image is returned as an `` tag suitable for direct inclusion in a web page. Here is the output:

```

<?xml version="1.0" encoding="utf-8"?>
<queryresult success="true" error="false" numpods="7" timing="0.357" timedout="0">
  <pod title="Input" position="100" error="false" numsubpods="1">
    <subpod>
      <plaintext>pi</plaintext>
      
    </subpod>
  </pod>
  <pod title="Decimal approximation" position="200" error="false" numsubpods="1">
    <subpod>
      <plaintext>3.141592653589793238462643383279502884197169399375105820...</plaintext>
      
    </subpod>
  </pod>
  <pod title="Property" position="300" error="false" numsubpods="1">
    <subpod>
      <plaintext>pi is a transcendental number</plaintext>
      
    </subpod>
  </pod>
  <pod title="Continued fraction" position="400" error="false" numsubpods="1">
    <subpod>
      <plaintext>[3; 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 2, 1, 84, 2, 1, ...]</plaintext>
      
    </subpod>
  </pod>
  <pod title="Alternative representation" position="500" error="false" numsubpods="3">
    <subpod>
      <plaintext>e^(pi i) = -1 (Euler's identity)</plaintext>
      π i</sup>=-1 (Euler's identity)" title="e<sup>π i</sup>=-1 (Euler's identity)" width="101" height="49"/>
    </subpod>
    <subpod>
      <plaintext>e^(2 pi i) = 1</plaintext>
      2 π i</sup>=1" title="e<sup>2 π i</sup>=1" width="55" height="30"/>
    </subpod>
    <subpod>
      <plaintext>e^(pi i k) = (-1)^k for k element Z</plaintext>
      π i k</sup>=(-1)<sup>k</sup> for k∈Z" title="e<sup>π i k</sup>=(-1)<sup>k</sup> for k∈Z" width="142" height="30"/>
    </subpod>
  </pod>
  ... remaining pods deleted for brevity ...
  <assumptions>
    <assumption>
      <word>pi</word>
      <categories>
        <category>NamedConstant</category>
      </categories>
    </assumption>
  </assumptions>

```

```

        <category>Character</category>
        <category>Movie</category>
        <category>Word</category>
      </categories>
    </assumption>
  </assumptions>
</queryresult>

```

Complete descriptions of the various query parameters and the result XML tags are given in later sections, but it can be seen that the output is a series of pods with subpods. The <assumptions> section at the end tells you what assumptions Wolfram|Alpha made about certain parts of the input, and what the alternative values are for each assumption. You could use this information to resubmit a query using a different assumption about the meaning of pi.

Sample Query with HTML+JavaScript Output

Callers who want to perform their own interpretation, transformation, or formatting of Wolfram|Alpha output might want the plaintext-and-image output demonstrated in the query in the previous section. If you want to embed Wolfram|Alpha results directly into your web page, with the same formatting and behavior (e.g., JavaScript buttons) that appear on the Wolfram|Alpha site, you can request results in HTML format.

Note that using the API in this way is for developers who want precise control over the type of results returned, or who want to make some modifications to the returned HTML. If you just want to put a Wolfram|Alpha search box on your web page, this can be more easily done with a prepackaged JavaScript control. See <http://www.wolframalpha.com/ad-toyoursite.html> for more information.

Here is the same sample query with HTML output:

```
http://api.wolframalpha.com/v1/query?input=pi&format=html&appid=xxxxx
```

Here is the result:

```

<?xml version="1.0" encoding="utf-8"?>
<queryresult success="true" error="false" numpods="7" timing="0.357" timedout="0">
  <pod title="Input" position="100" error="false" numsubpods="1">
    <markup>
      <div id="pod_0100" class="pod ">
        <hr class="top" />
        <h1><span>Input:</span><a href="javascript:showmathpop('Pi');" id="mathsource">
          <i>Mathematica</i> form</a>
        </h1>
        <div id="subpod_0100_1" class="sub">
          <div class="output" id="scannerresult_0100_1">
            
            <div class="annotpod">
            </div>
          </div>
        </div>
        <hr class="bot" />
      </div>
    </markup>
  </pod>
  <pod title="Decimal approximation" position="200" error="false" numsubpods="1">
    <markup>
      <div id="pod_0200" class="pod ">

```

```

        <hr class="top" />
        <h1><span>Decimal approximation:</span>
        <a href="javascript:asynchronousPod('pod.jsp?id=MSP22608468_1&ms=20',
'0200','','','0)"
            id="substitute_0200_0">More digits</a>
        </h1>
        <div id="subpod_0200_1" class="sub">
            <div class="output" id="scannerresult_0200_1">
                
                <div class="annotpod">
                    </div>
            </div>
        </div>
        <hr class="bot" />
    </div>
</markup>
</pod>
<pod title="Property" position="300" error="false" numsubpods="1">
    <markup>
        <div id="pod_0500" class="pod loading">
            <hr class="top"/><h1></h1><hr class="bot"/>
        </div>
    </markup>
</pod>
... remaining pods deleted for brevity ...
<scripts>
    <script src="/common/javascript/jquery/core/1.3.1/jquery.js" type="text/javascript"></script>
    <script src="/Calculate/javascript/init-min.js" type="text/javascript"></script>
    ... more script elements deleted for brevity ...
</scripts>
<css>
    <link href="/Calculate/css/Calculate-20090404-concatenated-min.css"
rel="stylesheet" type="text/css"/>
    <link href="/Calculate/css/ie-min.css" rel="stylesheet" type="text/css" media="screen"/>
</css>
<assumptions>
    <assumption>
        <word>pi</word>
        <categories>
            <category>NamedConstant</category>
            <category>Character</category>
            <category>Movie</category>
            <category>Word</category>
        </categories>
    </assumption>
</assumptions>
</queryresult>

```

In HTML format results, each `<pod>` element contains a `<markup>` element instead of `<subpod>` elements. This is because the pod is the most natural unit at the HTML level. HTML output makes extensive use of JavaScript. Many pods have JavaScript buttons that perform actions like displaying a popup window (the "*Mathematica* form" button in the first pod), or replacing the pod contents with a modified version (the "More digits" button in the second pod).

The JavaScript definitions for this functionality come in the `<scripts>` element. There is also a `<css>` element that includes `<link>` elements for the CSS files needed to duplicate the formatting seen on the Wolfram|Alpha site.

Of particular importance in this output is the third pod, titled "Property." Note that it is missing its content, and its outer `<div>` element has `class="pod loading"`. This is called an asynchronous pod, meaning that its content is swapped in later by an AJAX-style JavaScript call. The definition for the required JavaScript is in the `<scripts>` section (that particular script is not shown above for brevity). When providing HTML format results, Wolfram|Alpha by default allows pods that take longer to compute to be requested asynchronously, so that some results can be returned as quickly as possible. As described below, you can use the `async` query parameter to turn off this behavior, requiring all pods to be ready before the result is returned.

A typical way to use this output is to include the contents of the `<markup>`, `<scripts>`, and `<css>` elements in an `iframe` on your web page.

Query Parameters

There are numerous parameters you can specify in the URL to control how the query is performed and what types of results it returns. Here is an example request URL that uses several of these parameters:

```
http://api.wolframalpha.com/v1/query?input=5%20largest%20countries&appid=xxxx&podtitle=Result%20by%20are  
a&format=plaintext&timelimit=3
```

input

Specifies the input string, such as "5 largest countries". Of course the text must be URL-encoded, making this example string "5%20largest%20countries".

Either input or parseresult must be specified.

appid

An id provided by Wolfram Research that identifies the application or organization making the request.

Required.

podtitle

Specifies a pod title. You can specify more than one of these elements in the query. Only pods with the given titles will be returned.

Optional, default is all pods.

podindex

Specifies the index of the pod(s) to return. This is an alternative to specifying pods by title using the pod parameter. You can give a single number or a sequence like "2,3,5".

Optional, default is all pods.

format

The desired result format(s). Possible values are image, html, pdf, plaintext, minput, moutput, mathml, expressionml, and xml. See the "Output formats" section for descriptions of each of these types. To request more than one format type, separate values with a comma: "plaintext,minput,image".

Optional, defaults to "plaintext,image".

timelimit

The number of seconds to allow Wolfram|Alpha to compute and prepare results. When Wolfram|Alpha processes a query, it fires off a number of computations in parallel, called "scanners", each of which tries to produce a different category of output—a pod of its own. Some of these scanners return quickly and others take longer. To prevent output from being stalled by long-running scanners, Wolfram|Alpha supports a time limit beyond which scanners that have not finished are reported as timed-out. Timed-out scanners will not produce any pods in the output. The <queryresult> tag that wraps the entire result has a "timedout" attribute that gives the count of scanners that timed out. If this number is high, or if you are willing to wait longer to ensure the largest possible set of results, use the timelimit parameter to increase the time that scanners are allowed to work. Conversely, if you know that the pod(s) you want will be computed quickly, you can decrease the timelimit parameter so as not to wait for longer pods.

Optional, defaults to 3.

allowcached

In some cases, Wolfram|Alpha can return a cached result from an earlier query. Output often depends on up-to-the-minute information, or the locale of the computer making the query, so cached results are available less often than one might suspect. If you want to ensure that you do not get a cached result, specify false for this parameter.

Optional, defaults to true.

async

When delivering HTML+JavaScript output (the "html" result format type), Wolfram|Alpha can use an asynchronous mode that allows the result to come back before all the pods are computed. An example of this can be seen in the earlier sample query showing HTML+JavaScript output: the pod titled "Property" does not have its full HTML content present in the output. Instead, it has a <div> element of class "pod loading". When this HTML is loaded into a web browser, this <div> element will be replaced in an AJAX-like operation by a JavaScript function that is part of the <scripts> section further down in the results (the actual <script> tag containing this function is not shown in the output for brevity). This allows you to get some results into a web page very quickly, with longer-running pods being replaced with their content asynchronously. Specify false for the async parameter to turn off this asynchronous capability. In that case, you will get no result back until all the pods have been fully computed (still subject to the time limit provided by the timelimit parameter, however).

Optional, defaults to true. This parameter is only available for queries that specify the html result format type. For all other result types there is no asynchronous capability.

location specifications (ip, latlong, location)

Some Wolfram|Alpha computations take into account the caller's current location. By default, Wolfram|Alpha attempts to determine the caller's location from the IP address, but you can override this by specifying location information in one of three forms. The `ip` parameter lets you set the IP address of the caller, so if you are forwarding calls from your own web visitors to the Wolfram|Alpha API, you can propagate their IP addresses. The `location` parameter lets you specify a string like "Los Angeles, CA", or "Madrid", and the `latlong` parameter lets you specify a latitude/longitude pair like "40.42,-3.71". Negative latitude values are South, and negative longitude values are West.

Optional, defaults to determining location via the IP address of the caller.

assumption

It is often the case that a word or words in a query can have more than one interpretation. In the output for the query "pi", we see that Wolfram|Alpha has information about this word in the categories NamedConstant, Character, Movie, and Word. It chooses NamedConstant, but you can request an alternative assumption. The exact format for assumptions in queries and results is not yet defined.

Optional.

units

Lets you specify the preferred measurement system, either "metric" or "nonmetric" (U.S. customary units).

Optional, defaults to making a decision based on the caller's geographic location.

currency

Lets you specify the preferred currency unit. Wolfram|Alpha supports a very large set of currency types, including (names are case-insensitive): USDollars, Euros, BritishPounds, SwissFrancs, CanadianDollars, AustralianDollars, BrazilianReals, ArgentinePesos, MexicanPesos, RussianRubles, IndianRupees, HongKongDollars, SingaporeDollars, KoreanWon, TaiwanDollars, Yen, and ChineseYuan.

Optional, defaults to making a decision based on the caller's geographic location.

moreoutput

Some pods have both a long and short set of subpods to display. For example, in the screen shot of the Wolfram|Alpha site output for "pi" at the beginning of this document, the "Alternative representation" pod has three subpods and a "More" button. If a user clicks the "More" button, the pod is replaced by a longer list of subpods showing more identities involving pi. The `moreoutput` parameter lets you specify whether you want the result to contain longer or shorter lists of subpods. The shorter lists are what Wolfram|Alpha considers to be the most important results, and longer lists could take a bit longer to generate. Specify true for `moreoutput` to request long lists.

Optional, defaults to false.

width

The page width in pixels for which the output should be formatted. Width adjustment is an experimental feature. Wolfram|Alpha results are carefully optimized for the default width, and changing it significantly might interfere with proper formatting.

Optional, defaults to 500.

parseresult

The `validatequery` API function, described later, allows you to call Wolfram|Alpha to quickly determine whether it can understand a given input. It returns a result that represents the internal form of the query after the parsing stage is finished. You can pass this intermediate expression into the `query` function to get the final result without having to redo the parsing stage. You do this via the `parseresult` parameter. Set it equal to the contents of the `<parsedata>` element obtained in the result from `validatequery`. Do not use the `input` parameter when you are using `parseresult`, as `parseresult` is an alternative means of supplying the input.

Optional, alternative to the `input` parameter.

Result Description

Results come back in XML format. Here is a listing of the result elements and their attributes.

<queryresult>

`<queryresult>` is the outer wrapper for all results from the `query` function. It has the following attributes:

success	true or false depending on whether the input could be successfully understood. If false there will be no <pod> subelements.
error	true or false depending on whether a serious processing error occurred, such as a missing required parameter. If true there will be no pod content, just an <error> subelement.
numpods	The number of pods.
timedout	The number of pods that are missing because they timed out (see the <code>timelimit</code> query parameter).
timing	The wall-clock time in seconds required to generate the output.

<pod>

<pod> elements are subelements of <queryresult>. Each contains the results for a single pod. <pod> has the following attributes:

title	The pod title, used to identify the pod.
error	true or false depending on whether a serious processing error occurred with this specific pod. If true, there will be an <error> subelement.
position	A number indicating the intended position of the pod in a visual display. These numbers are typically multiples of 100, but it is only relevant that they form an increasing sequence from top to bottom.
numsubpods	The number of subpod elements present.

<subpod>

<subpod> elements are subelements of <pod>. They do not appear if the requested result format is html. Each contains the results for a single subpod. <subpod> has no attributes.

<plaintext>

<plaintext> elements are the textual representation of a single subpod. They only appear if the requested result format is plaintext. <plaintext> has no attributes or subelements.

<minput>

<minput> elements are the *Mathematica* input representation of a single subpod. They only appear if the requested result format is minput. <minput> has no attributes or subelements.

<moutput>

<moutput> elements are the *Mathematica* output representation of a single subpod. They only appear if the requested result format is moutput. <moutput> has no attributes or subelements.

<mathml>

<mathml> elements enclose the Presentation MathML representation of a single subpod. They only appear if the requested result format is mathml. <mathml> has no attributes.

<expressionml>

<expressionml> elements enclose the ExpressionML representation of a single subpod. They only appear if the requested result format is expressionml. <expressionml> has no attributes.

 elements are HTML `img` elements suitable for direct inclusion in a web page. They point to stored image files giving a formatted visual representation in a single subpod. They only appear if the requested result format is img.

<markup>

<markup> elements are subelements of <pod>. They only appear if the requested result format is html. Their content is the HTML for an entire pod. <markup> has no attributes.

<scripts>

The <scripts> element is a subelement of <queryresult>. It will only appear if the requested result format is html, and there will only be one. Its content is a series of <script> elements defining JavaScript functionality needed by the HTML in the <markup> elements. <scripts> has no attributes.

<css>

The <css> element is a subelement of <queryresult>. It will only appear if the requested result format is html, and there will only be one. Its content is a series of <link> elements defining CSS files needed by the HTML in the <markup> elements. <css> has no attributes.

<assumptions>

The `<assumptions>` element is a subelement of `<queryresult>`. Its content is a series of `<assumption>` elements. It has no attributes.

<assumption>

The `<assumption>` element is a subelement of `<assumptions>`. It defines a single assumption about the meaning of a word or phrase, and a series of possible other values. Its content is structured like the following example:

```
<word>pi</word>
<categories>
  <category>NamedConstant</category>
  <category>Character</category>
  <category>Movie</category>
  <category>Word</category>
</categories>
```

The first-listed category is the currently assumed value. The other categories represent possible alternatives. You could use these values to create a means for your users to redo a query with a different assumption.

<error>

The `<error>` element occurs as either a subelement of `<queryresult>`, if there was a failure that prevented any result from being returned, or as a subelement of `<pod>`, if there was an error that only prevented the result from a given pod from being returned. `<error>` has the following attributes:

code	The error code, an integer.
msg	A short message describing the error.

The validatequery Function

So far we have been dealing with the `query` function. There is another function in the Query API, called `validatequery`. This is a specialized function that performs only the initial parsing phase of Wolfram|Alpha processing. Its purpose is to quickly determine whether Wolfram|Alpha can make sense of a given input, bypassing the more time-consuming stages of fully analyzing the input and preparing results. Here is a sample URL that calls the `validatequery` function:

```
http://api.wolframalpha.com/v1/validatequery?input=pi&appid=xxxxx
```

Here is the result:

```
<?xml version="1.0" encoding="utf-8"?>
<validatequeryresult success="true" error="false" timing="0.080">
  <parsedata>
    1:eJxTTMoPymRhyGAAAtU (...remaining characters deleted for brevity)
  </parsedata>
  <assumptions>
    <assumption>
      <word>pi</word>
      <categories>
        <category>NamedConstant</category>
        <category>Character</category>
        <category>Movie</category>
        <category>Word</category>
      </categories>
    </assumption>
  </assumptions>
</validatequeryresult>
```

The success attribute of the `<validatequeryresult>` element tells you whether Wolfram|Alpha could successfully parse the input string. If its value is true then it is highly likely that Wolfram|Alpha can return sensible results for this input using the `query` function. The `<parsedata>` element contains an encoded representation of the parsed intermediate form of the query. You can pass this value as the `parseresult` parameter in a subsequent call to the `query` function to avoid having to reparse the input.

Note that `<parseresult>` can contain `<assumptions>` information. This information is available at the close of the parsing step, so it is provided in the result. You can use it in the same way as you would from the result of the `query` function, such as to perform a subsequent `query` call with a modified assumption.

“Data” API

Wolfram|Alpha has a simple yet expressive internal mechanism for extracting specific pieces of data from its enormous store of curated data. In the near future we will be exposing this facility via a web API. This type of request can be used when you want a single result value or set of values and can describe the data you want in a precise way (or at least ask Wolfram|Alpha to assist you in determining the correct way to ask). More information on this lower-level API will be available soon.