# MAXXCAT

## A Quick Glance at the MaxxCAT API

The MaxxCAT API has been designed to be as simple as possible, with the ancillary benefits that our compact API and data protocol transmit very quickly over the Internet, and are immediately usable by browsers and other applications. Unlike XML and other overly complex, overly verbose data structures, MaxxCAT by virtue of JSON and HTTP delivers a lean, fast, crisp interface to the high performance MaxxCAT enterprise search appliances, with a minimal learning curve for the developer.

Applications that do search generally need to do three things -- crawl data sources, query for matches, and disseminate results to either a User Interface or application code that is using a MaxxCAT appliance for high performance query resolution. After reading this two page paper, most developers will have a sufficient understanding of how the API operates to begin building applications.

### Crawling Data Sources

MaxxCAT appliances are controlled by .collection files that set up all of the parameters of a given data source, including how to access the source, when and how frequently to access the source and any security information that is either needed to access the source, or that will be required of users to view the data. Here is a basic configuration file, that tells the MaxxCAT appliance to crawl the www.maxxcat.com website, weekly, and to place all of the contents of the website into a logical segment, or COL_ID, of 10. The COL_ID can be used at other points in the API to separate or to facet search results and implement security.

*URL=http://www.maxxcat.com COL_ID=10*
*CONFIG.CRAWL_INTERVAL=WEEKLY*

### Sending A Query

Retrieving search results from the appliance uses a simple HTTP call, of the form

*http://queryserver.maxxcat.com/query.cgi?query=maxxcat&collection=colid10*

This query indicates to the appliance that it is to retrieve a results set, or list of documents, that contain the term "maxxcat", and which exist in a collection with a COL_ID of 10.

Boolean queries are possible, in the form

*http://queryserver.maxxcat.com/query.cgi?query=maxxcat+appliance*

In this case, any document on the appliance, regardless of the COL_ID will be returned, so long as it contains both terms, maxxcat and appliance.

One of the excellent features of MaxxCAT is its commitment to performance, and the API supports multiple, parallel queries in the same call, that will produce multiple results sets. In this case, the NQ (New Query) operator is used, to indicate to the appliance that both queries should be done on the same connection, thus eliminating connection set up time for multiple queries.

*http://queryserver.maxxcat.com/query.cgi?query=maxxcat&collection=colid10&NQ&query=maxxcat+appliance*

### Interpreting the Results

Results that come back from the appliance are by default, standard JSON objects of the form:

```
callBack({
  "resSet":[{
    "results":
    [
      {
        "res":
        {
          "url":"http://www.yoursite.com/manual.html",
          "title":"Instruction Manual for Widget X",
          "snippet":"Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor incididunt ut labore
et"
        }
      },

      {
        "res":
        {
          "url":"http://www.yoursite.com/about.html",
          "title":"About Our Company",
          "snippet":"Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut"
        }
      },

      {
        "res":
        {
          "url":"http://www.yoursite.com/contact.html",
          "title":"Contact Us!",
          "snippet":"xcepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id
est laborum"
        }
      }
    ],

    "resStart":"0",
    "resEnd":"3",
    "total":"543"
  }],

  "time" : "0.415950"
})
```

Applications or enterprise search solutions can use the emitted JSON in its raw form, or use JavaScript or any other programming language to massage the results into the desired final form. MaxxCAT appliances ship with a standard default template that includes all of the JavaScript necessary to quickly build search results templates in HTML, that are completely and simply customized to a users requirements.

www.maxxcat.com

# MAXXCAT

## A Quick Glance at the MaxxCAT API

### Setting up a Database Crawl

In addition to using .collection files to specify data sources like web sites, file servers, applications etc, by using the MaxxCAT BobCAT SQL connector, virtual documents can be created from data bases very quickly, and high speed searches of the full text, and unstructured data can be performed. The files that specify database interaction are known as .frm files. Once MaxxCAT has crawled a database, the query load, no matter how heavy it is, will not impact the underlying database as the MaxxCAT does all the work.

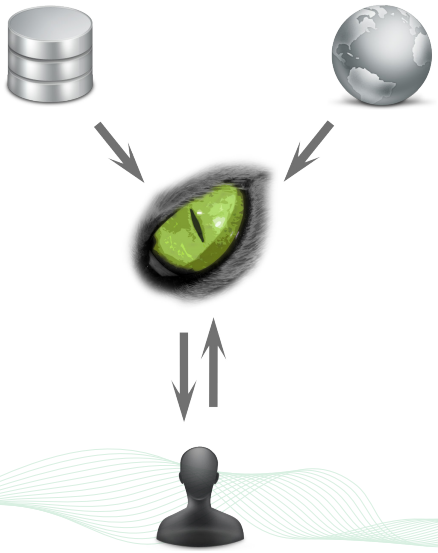
A simple example would be to query an Order History database for all orders and line items so that a customer service department could use a simple web interface to look up orders by customer name (first or last), zip code, or any word that would be in any description of an item on the order.

*SELECT ORDER_NUMBER, ZIP_CODE, BILL_LASTNAME, DESCRIPTION, QUANTITY FROM ORDERS,DETAILS WHERE ORDER.ORDER_NUMBER=DETAILS.ORDER_ NUMBER*

With this simple SQL statement, MaxxCAT will create a record of each order in the database, and will return in the JSON results either the full text of the record, or sufficient information about the record to allow an application to call the database directly for more information.

*http://queryserver.maxxcat.com/query.cgi?query=16202*

This would find all orders that had an order number of 16202, a zip code of 16202 or any item that has the description of 16202, and return them to the user interface for further action.



### Controlling the Device

In more advanced applications, MaxxCAT search appliances may be used as subsystems or compute servers, and the user may desire to place MaxxCAT under the control of another computer. This may be desirable if certain events in the application may require behavior from the MaxxCAT search appliance that depends upon events happening elsewhere in the application.

Other computers or programs can control MaxxCAT search appliances through the standard MaxxCAT API, which provides calls for initiating crawls, indexing collections, manipulating data and appliance status. Consider that a newspaper publisher may want to create transient collections with articles that were published in the last 24 hours, and then to destroy and replace the collection whenever an editor, using an external publishing system, marks a new deadline for a virtual edition. How would the publishing system signal to the MaxxCAT to update the data for the recent articles collection?

*http://queryserver.maxxcat.com/maxxcat.cgi?resetcollection+recentarticles*
*http://queryserver.maxxcat.com/maxxcat.cgi?hotcrawl+recentarticles*
*http://queryserver.maxxcat.com/maxxcat.cgi?reindexall*

Another example would be to institute a crawl that appends a database driven collection every time a new article is submitted to the system. In this case, the collection control file configures the collection for append mode, which just adds records to the collection upon each crawl, rather than recrawling the entire data set. So a database trigger in the publishing application may want to notify the maxxcat that new data is available at the instant that it is committed to the publishing database.

*http://queryserver.maxxcat.com/maxxcat.cgi?hotcrawl+allarticles*
*http://queryserver.maxxcat.com/maxxcat.cgi?reindexall*

Hopefully this quick overview of the API will have given the reader a sense of the simplicity and power of the API to talk to numerous data sources and to handle the simple, structured JSON output. The MaxxCAT API is intuitive and terse, allowing developers programmatic access to functions on the search appliance that range from standard full text search queries, to database integration and direct device control. Additional detailed information can be found in the MaxxCAT User Guide.