NAB K2 SharePoint CAML / Search Broker

# Overview

This broker provides a method to craft optimized queries that will run against a SharePoint farm and expose them as specific broker objects to K2 Blackpearl. This broker provides different functionality than the out of the box broker in that queries are pre-defined and optimized at design time so as to provide runtime performance that is more finely tuned to the need of a K2 form or workflow.

This broker can be described as providing “views” into your SharePoint farm just as you might create highly optimized views or stored procedures in Microsoft SQL Server to only return columns needed for an operation. So too can you create those CAML / KQL (and FQL) queries using this broker to return just the data you need.

You can find this document and full code at: https://github.com/thystonius/NAB.K2

## General Features

* Fine grained control over which columns are returned, advanced filtering, and sorting when retrieving information from SharePoint
* Ability to specify arbitrary and calculated parameters to a query
* Ability to engage SharePoint Search with full support for KQL (as is allowed in your SharePoint environment)
* Ability to pull Refiner values for a query to provide users with all the functionality of a standard SharePoint search
* Ability to cache certain query results to reduce SharePoint load and improve speed on infrequently changing list
* Query execution based on different credentials to provide controlled access to lists where it is not desirable to provide direct SharePoint access
* Publish configuration files with zero changes between development and production
* Provides a way for SharePoint administrators to review exact CAML and KQL queries before being put into production

## WARNING

The purpose of this broker is to allow trained SharePoint developers / engineers to boost the performance of the K2 environment by using highly optimized queries and searches. However, this broker, as with many others, can be configured to abuse its data source. There are numerous internet resource on how to optimize CAML, KQL queries, and properly configure SharePoint that will allow you to make significant performance gains by using this tool properly. Please be cautions not to abuse the SharePoint Query and Search interfaces by avoiding:

Returning large result sets (many rows or many columns)

Perform inefficient CAML queries against unindexed columns or in arbitrary order (remember filter order matters to CAML)

Returning data that is not required

Not enabling caching when possible

Overusing Impersonation for data that is accessible to all users

Additionally do not abuse your K2 infrastructure by:

Cache large data sets for extended periods

Enabling cache for queries that contain wide ranging parameter, time based parameters, or calculated parameters that would pollute the cache

Overusing the internal K2 paging, searching, and sorting facilities on large data sets

## Use Cases

### Optimized CAML Query

* + Creating a CAML query that is specific to your need in either a Workflow or SmartForm will provide better performance by including only the data that is needed for the operation.
  + Provide a data source for a SmartForms Lookup or Drop-Down control and would like to use a column other than the ID for fast lookups or where the List may contains many columns and the need is only for a small subset.
  + Provide server based sorting on returned results.
  + Add filters to CAML queries such as “between”, “starts with”, or more complex AND / OR logic to reduce the records returned to only what is needed.
  + Test and profile CAML queries prior to deployment to ensure consistent performance between dev and production.

### Caching

* + Server based caching of the high volume list information to reduce round trip to SharePoint for information.

### Security Shifting

* + By executing a query using different credentials can allow controlled access to lists where it is not desirable to provide direct SharePoint access.
  + Create a CAML query that only return the “public” viewable columns in a list such as HR information where the list itself remains inaccessible to users.
  + Using a single security credential (such as Service Credentials) for high volume queries reduces the number of authentication / impersonation handshakes between K2 and SharePoint

### Search Refiners

* + Using a pure refiner search you can display a list of all the Content Types (or any refine-able field) that meet a search query. For example, display a list by Content Type of all items created by a certain user. Then use those refiner values to display just those search results.

### Utilize SharePoint Search

* + Take advantage of SharePoint Search via K2 Smart Objects to retrieve related information.

## Designed for, but does not yet support

The broker is designed for these features, but will be implemented in a future release.

### Paging & Sorting

* + This broker does not yet support native paging and sorting.

### OAuth

* + This broker does not yet support OAuth.

### Calculated Parameters

* + Simplify some CAML queries by using a calculated value instead of passing in a value from K2. Particularly useful for complex Date math that might not be supported by CAML

### Post Calculation

* + Some values from SharePoint can be difficult to parse in K2 or could be parse with significantly more efficiency if done in the broker. Fields such as Users would be good candidates for post processing prior to returning to K2.

## Not Design For

These items are intentionally not part of the design and should be considered anti-use-cases of this broker.

### Updated

* + This broker does not provide any update capabilities to SharePoint. The out of the box K2 brokers work very well at all write operations.

### Designable at Runtime

* + Configuration and designing of queries for this broker cannot be done at runtime. All queries must be pre-configured and stored into a configuration file and then published into production. Not only does this encourage a good change management processes, but also ensures that users are not able to re-craft or change queries at runtime (other than what the designer had intended). Additionally in highly managed environments, these configuration files should be audited for performance and security concerns prior to deployment.

### CAML / KQL Designer

* + This broker or the configuration editor are not CAML or KQL design tools. There are plenty of those on the market that all work very well at helping you create those queries. This tool will allow you to hand type (and generate) the queries, test, and profile their use.

# Installation

This broker comes in two parts. The first is the actual broker used by K2 and the second is a small configuration and testing application that is used to create a configuration file used by the broker during runtime. These components can run on the same machine, however it is recommended that you install the Configuration Editor in a separate location so as to conform to K2 standards.

## Server Installation

The installation is identical to other K2 Service Brokers and only requires the DLL to be copied into the proper K2 folder and then registered.

### Prerequisites

K2 Blackpearl 4.6.11 (may work with earlier version, but no testing)

K2 Blackpearl 4.7 (compiled using Framework 4.6 and will be released when 4.7 is released)

SharePoint Client Components for SharePoint 2013

You must have SharePoint Search properly configured to be able to use Search queries. Configuration of SharePoint search is beyond the scope of ~~anyone’s lifetime~~ this document.

### Install

Copy NAB.K2.SharePointSearch.dll into the Service Broker folder:

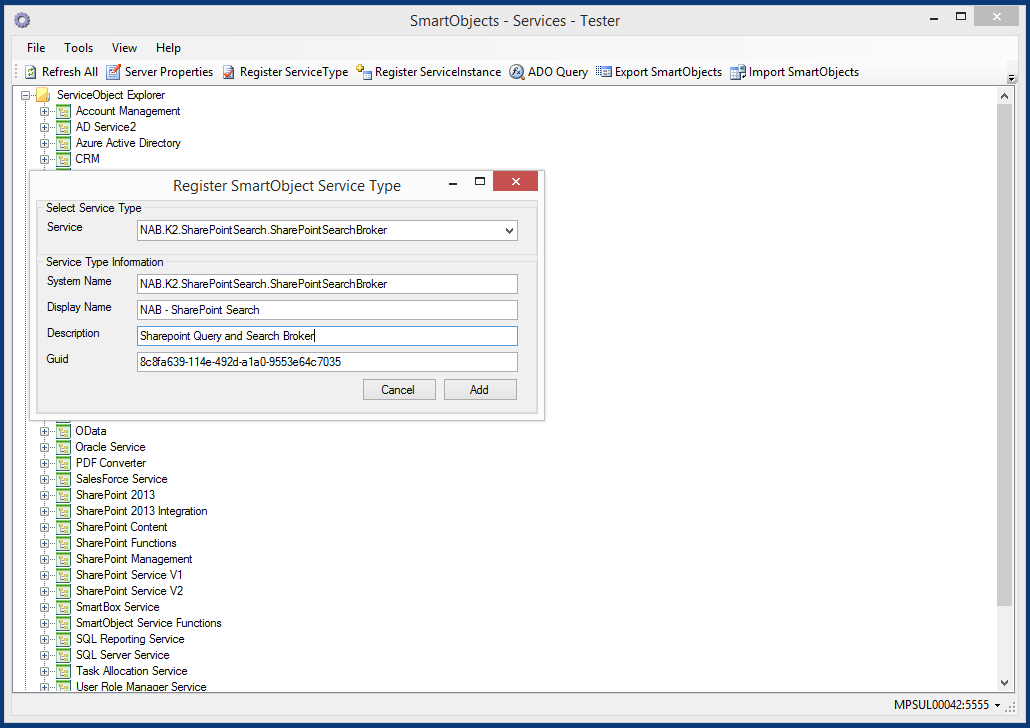
C:\Program Files (x86)\K2 blackpearl\ServiceBroker

Follow the standard Service Broker installation steps outline here:

<http://help.k2.com/onlinehelp/k2blackpearl/DevRef/4.6.11/default.htm#RegisteringCustomServiceBrokers.htm>

Restart your K2 Host Server to ensure the new Assembly is recognized

Using the Smart Object Service Tester, press the “Register Service Type” button.



Select the NAB.K2.SharePointSearch.SharePointSearchBroker server from the drop down box at the top. Then enter your own Display Name for this broker. It is recommended that you keep the System Name the same as shown above.

## Configuration Development Machine Installation

The Configuration Editor is a small window executable used to create the configuration file for use by the broker during runtime. K2 is not required to be installed on this machine.

### Prerequisites

SharePoint Client Components for SharePoint 2013

### Install

Create application folder where desired, for example C:\Program Files\NABK2SPSearchBroker

Copy files from Development Pack into folder

# Configuration Process Overview

This broker was designed to encourage a good change management process where design and development are separate from the publishing to production. This is achieved by requiring the publishing of a configuration file onto the production environment where it can then be used during runtime. Additionally, this configuration file contains no full URLs (those are configured in the service instance), thus a configuration file should not need to be changed between dev, test, and production.

## The Configuration File

This broker relies on a configuration file that defines all the queries that it will expose to the K2 environment for a single Service Instance. Those queries must be predefined by development or business team to meet both the needs of the particular solution but also to properly use the SharePoint and K2 resources. You may configure as many additional Service Instances each using different SharePoint URLs and authentication settings.

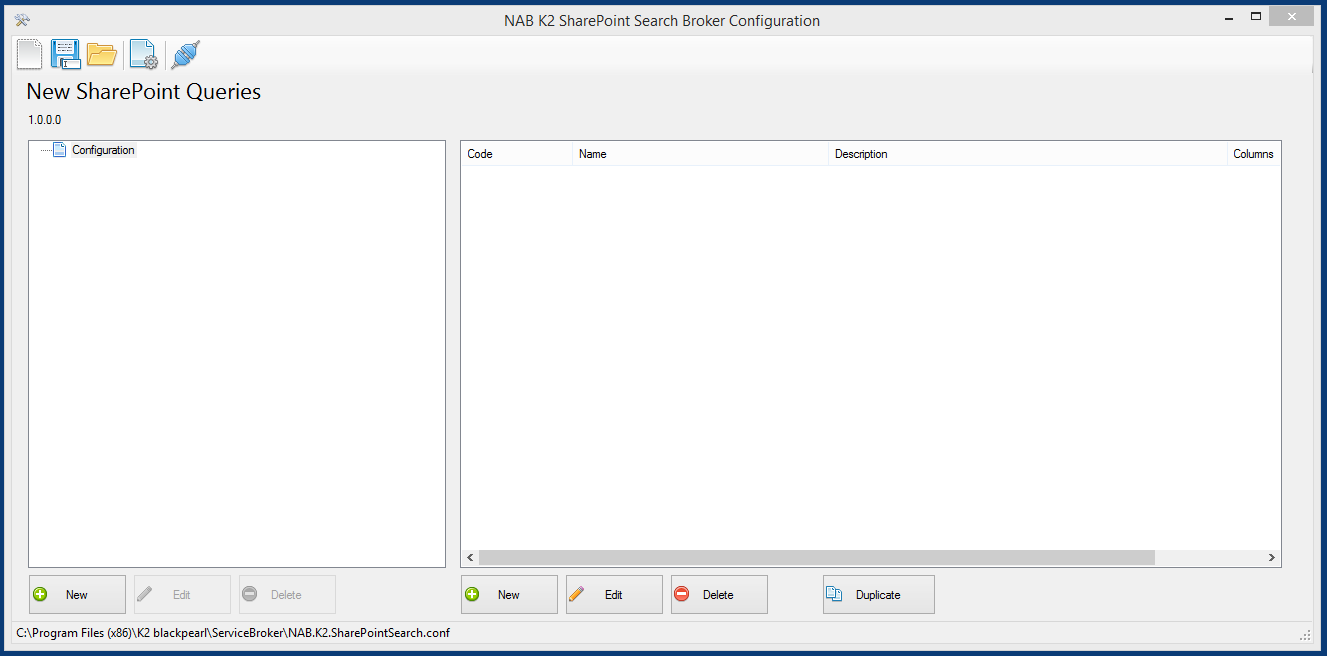
Queries within the configuration file contain only relative URLs to SharePoint, thus if your test environment mirror production then this will allow you to use the configuration file between dev and production.

It is recommended that you group your queries into logical sets that match their intended use and match the intended settings of the service instance. For example, the authentication setting is set per-service instance, thus if you have a group of queries that you want to execute using the “Service Account” then those queries should all go into a single configuration file.

During design time, this file can be placed anywhere accessible. During runtime, this time must be placed in a location that can be accessed by the K2 Host Server. It is possible to place the configuration file itself on SharePoint (more on that later).

### Configuration Editor

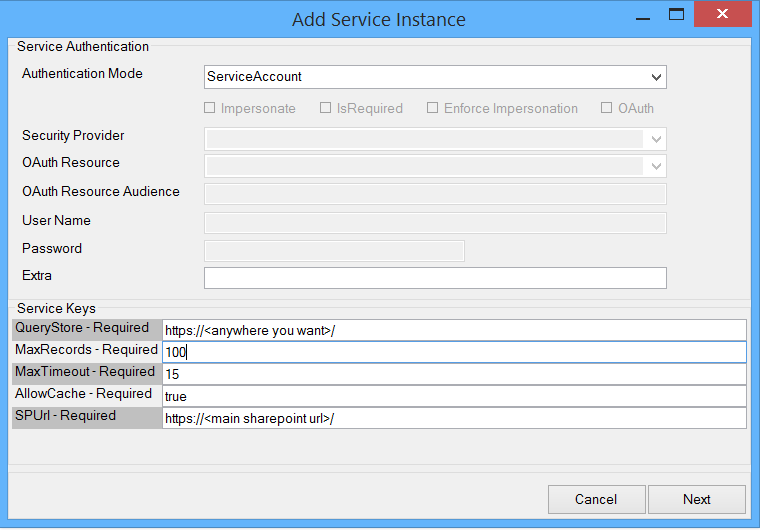
The Configuration Editor is a small Windows executable that is used to build a configuration file that is used by the broker as a definition of that it should provide to K2. This is located in a folder used during the Installation step. Please see the detailed guide on Building Queries later in this document for more information on building queries.



After you have finished building queries you must then save your configuration file to a location that will be accessible by your K2 Host Server.

## Registering Service Instance

As stated above, you must first choose a location for the configuration file that is accessible by the K2 Host Server (and the Service Account). For single server installations, the most logical place for this file to be located is on the K2 Server itself. For larger installations, a network share would be more practical as this must be accessible by all instances of the K2 Host Server.



Service Instance Configuration (Service Keys)

|  |  |
| --- | --- |
| Name | Description |
| QueryStore | Path or URL to configuration file.  Examples:  Path: **C:\Program Files (x86)\K2 blackpearl\ServiceBroker\NAB.K2.SharePointSearch.conf**  URL: **https://<some internal site>/K2SPQueries.conf**  Note that Service Account credentials are used to access both the files system and any remote site. Ensure that the service Account has proper access to the resource you are attempting to reach. Also it is not advisable to use a configuration file from outside your network.  Currently only http and https are supported for downloading (don’t even think of using ftp, really in this day and age!)  It is strongly encouraged that you Deny Write to the Service Account used by the K2 Host Server. This will ensure that there is no way for any K2 action to change this file. |
| MaxRecords | Maximum number of *records* that this service instance will ever return as a result from a single query. This is the absolute maximum and will override any values in a query configuration. |
| MaxTimeout | Maximum number of *seconds* that the broker will wait for a query to return before raising a timeout exception. This is the absolute maximum and will override any values in a query configuration. |
| AllowCache | (true / false) Indicating if caching is enabled for this service instance. Please read the section on caching for more information about how this functions. |

## Caching

This service broker can be configured to cache the results from queries and store them locally in memory on the K2 Server. This feature can be used to great effect to reduce the amount of data and number of queries executed against a SharePoint environment. However, when improperly used, this can also be used to significantly hamper performance by consuming memory on the K2 Server and still tax a SharePoint farm.

This broker currently does not support user-level caching, thus caching is only activating when the Authentication mode is set to Static or ServiceAccount.

### When to Use Caching

When the Service Instance is set to use Static or ServiceAccount credentials

When the known data set is small

When the data is going to be accessed repeatedly and often

When the data set does not need to be security trimmed for each user or where the trimming can easily be performed by K2

### When to NOT Use Caching

When the data set is known to change

This broker does not perform any mechanism other than time to invalidate cache records. So if the data is changed frequently or is known to change during the expect cache lifetime, then caching should not be used.

When the Service Instance is set to use Impersonation, SSO, or OAuth credentials

It is not possible to properly security trim query results in such a way that they can be re-used by different users, thus caching is disabled on any service instance with the above credentials. It is recommended to create a separate configuration file and separate service instance for all the queries that should run under different credentials.

When the known data set is large

Caching should not be used on data sets that contain large numbers or rows, columns, or contain large data sizes such as files. It is best to find a different way to provide this data rather than using this broker when this is required. This is certainly subjective to the resources in your environment.

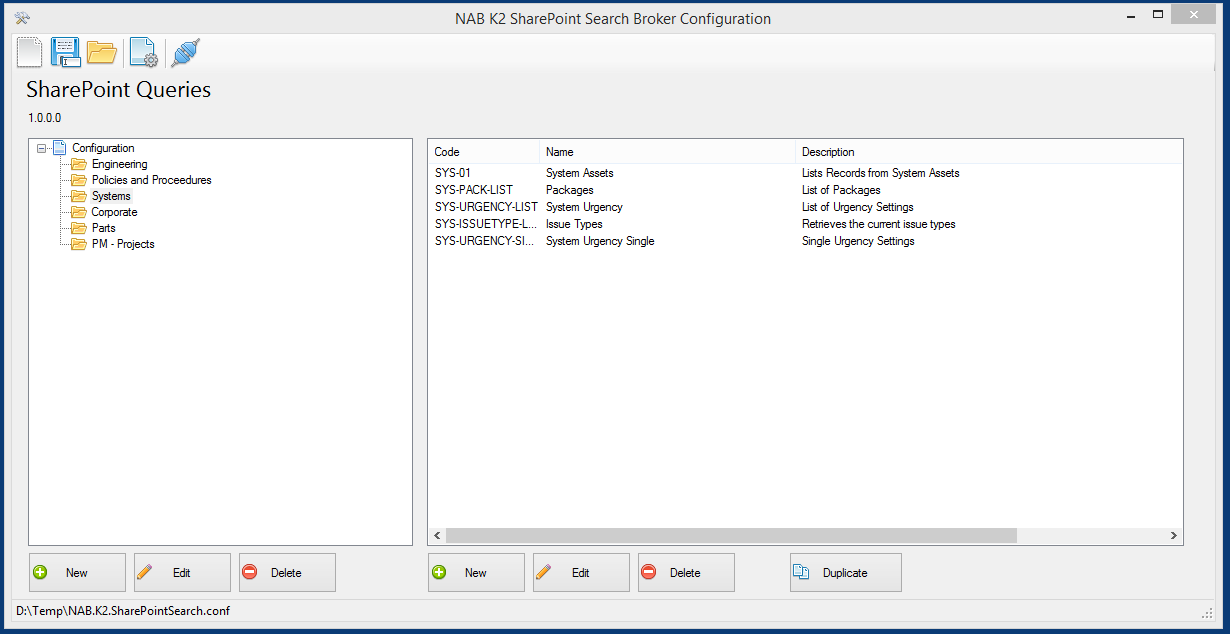
When the data is infrequently used

When it comes to timing, caching is a balance between the time and resource needed to request the data and the resources and overhead required to temporarily store it.

# Building Queries

The configuration editor is a very basic tool that allows you create “queries” that present a fixed set of inputs and outputs to K2, much like you would create Views or Stored Procedures in a database. By creating, testing, and profiling these queries in advance you make it easy to consume them from K2.

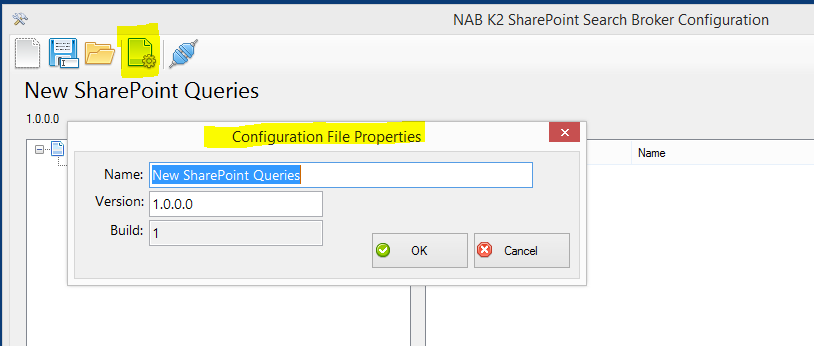
## Overview



The main window consist of basic toolbar buttons at the top, a list of folders on the left, then a list of queries on the right.

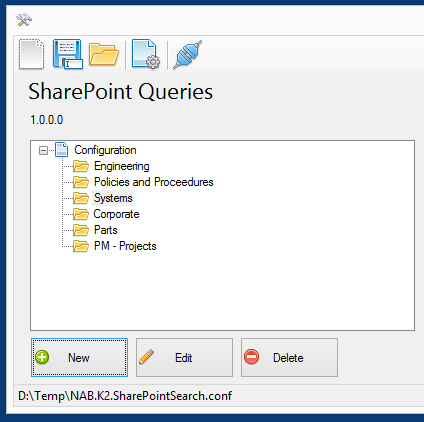
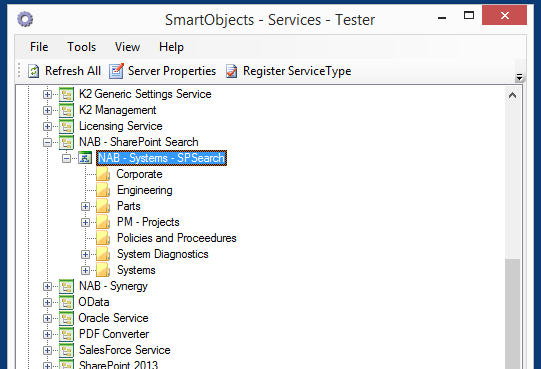
By default the editor will open the last file saved.

Pressing the Configuration File Properties toolbar button will display a small window allowing you to set the Name and provide a Version Number for this file. These values are for the you, as the designer’s, benefit and can be used in any way you see fit that will help organize your configuration files and keep track of changes. The Build number is a rolling integer number that increments ever time the file is saved. This serves no other purpose than to provide an automated change tracking number to help with any change management practices. Both the name and version number will be displayed in the main window when a configuration file is open.

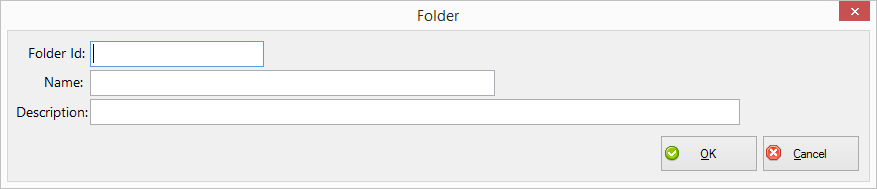


## Folders

All queries must be contained within a Folder. This is an arbitrary grouping of queries and serves no other purpose other than segmentation. These folders will be how the queries are displayed when this configuration file is used in K2 as show below:

Create folders by pressing the “New” button on the left hand side of the main window.

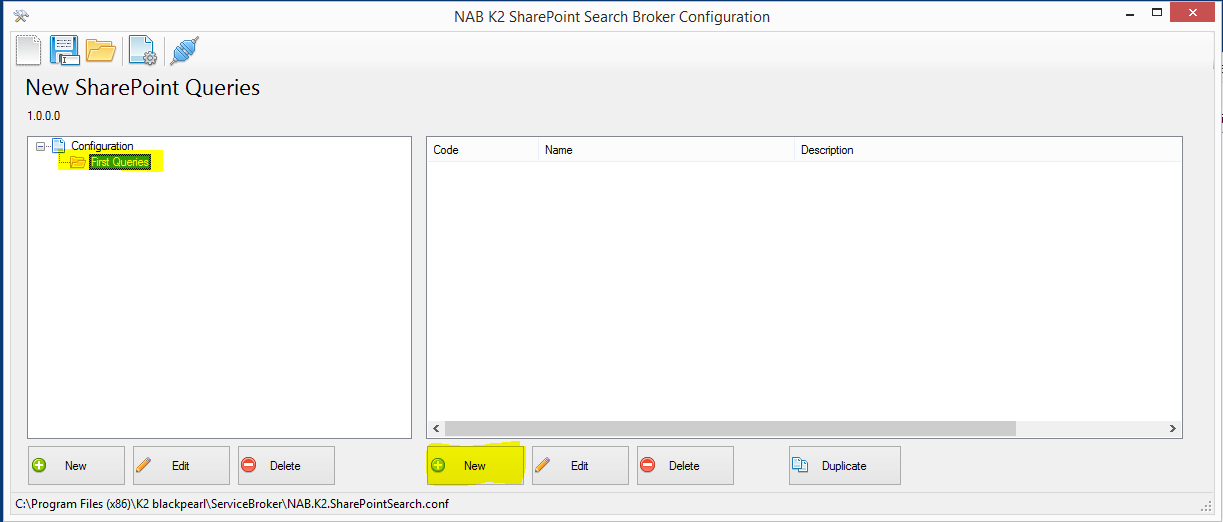


|  |  |
| --- | --- |
| Name | Description |
| Folder Id | The unique identifier for the folder within this configuration file |
| Name | The display name of the folder and what will be displayed in K2 designers |
| Description | Long description of the folder (this does show up somewhere in K2, but not many places) |

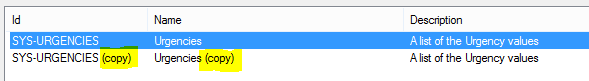
All fields of a folder can be edited after it is created. Deleting a folder will remove all its contents.

## Queries

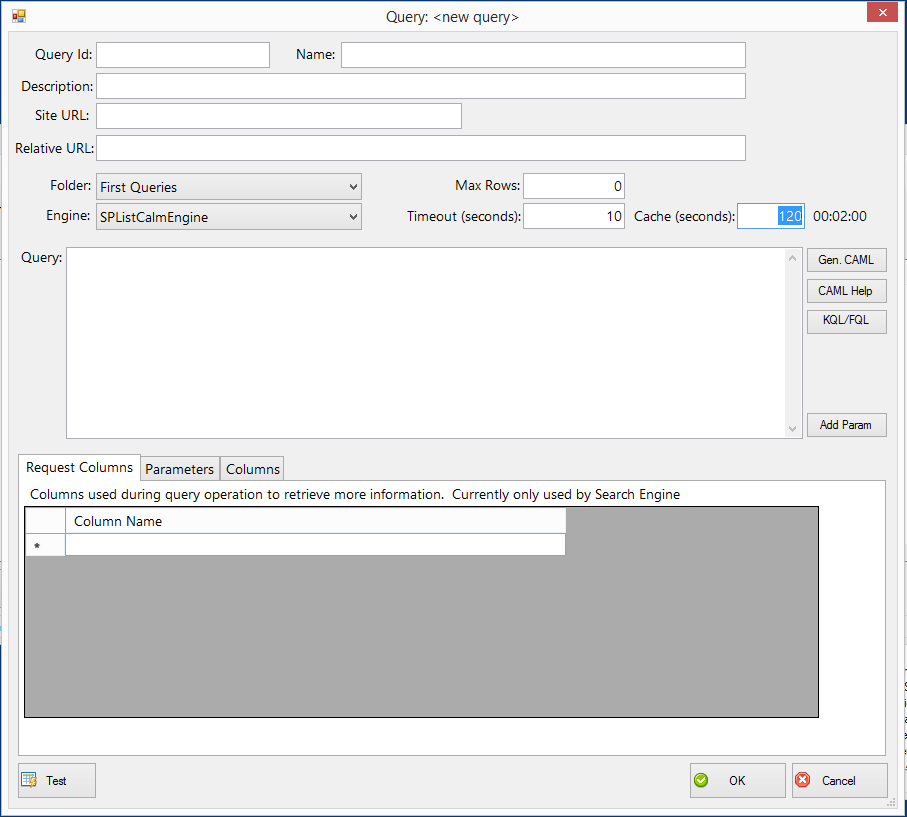
Create a new query by first selecting a folder then press the New button on the right hand second of the main window.



Additionally queries can be copied using the Duplicate button which will make a copy of the selected query. This can be useful to test small changes or to make a series of query for the same list but with different parameters or return columns. Each copy will have the words “copy” appended to both the Id and name.



### Query Editor Window

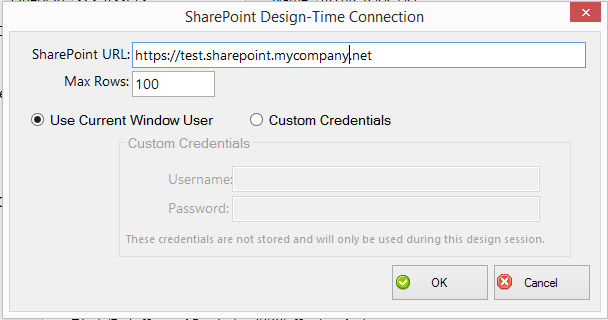


|  |  |
| --- | --- |
| Name | Description |
| Query Id | The unique identifier for this query within this configuration file. It is recommended to make this some kind of logical id as this will be used later for query profiling. |
| Name | The display name of the query and what will be displayed in K2 designers |
| Description | The long description of this query (displayed somewhere in K2) |
| Site URL | The relative URL of the Site where the query will be executed. This MUST be the site and not the full path to the Library or List. This is to make it easier for the broker to request information from the site (such as User information). |
| Relative URL | This is then the site relative URL to the list or library that will be used for the query. |
| Folder | The folder in the configuration file for this query. Changing this will simple moving the query within the configuration. |
| Engine | Specifies if this query will be a CAML query or use SharePoint Search (more on this below) |
| Max Rows | The maximum number of rows that this query will ever return. During runtime the actual maximum number of rows return is the MINIMUM of this value and the MaxRecords value specified by the Service Instance. |
| Timeout | The timeout for the query. If a query takes longer than this value, then a timeout exception will be thrown back to K2. |
| Cache (seconds) | The number of seconds that the return value for this query will be cached. A value of zero will disable caching. Please see the Caching section for more detail about how caching operates. |
| Query | The query expression in either CAML or (KQL, FQL) depending on the engine. |
|  |  |

There are thee tabs at the bottom of the query editor window. The Request Columns is used exclusively by the SharePoint Search Engine, the Parameters is used to add input parameters to a query, and the Columns tab allows for the configuration of the output columns of the query.

The query creation process starts with the filling in the basic information above, constructing a proper query text, then using the “Detect” button on the Columns tab to automatically detect the output columns.

During the design process anytime the editor attempts to make the first connection to SharePoint it will prompt you for the “base” URL for SharePoint along with your credentials. This will happen only once while the editor is running. The base URL is the same URL that you would enter during the configuration of the Service Instance as mentioned earlier in this document. This screen will save the SharePoint URL on your machine only (not into the configuration file). Your credentials are NEVER stored in the configuration file or in any settings file and are only used during the design time experience and only while the editor is running.

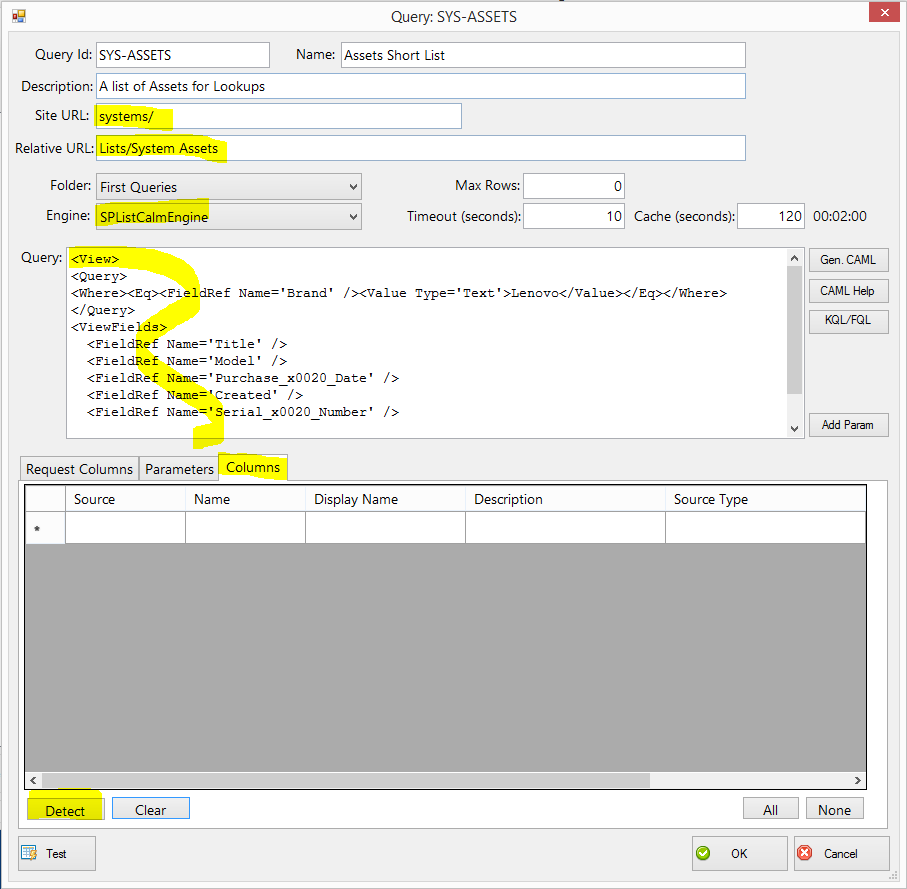


You can get back to this screen at any time by pressing the “Design Time Connection” button in the toolbar of the main window.

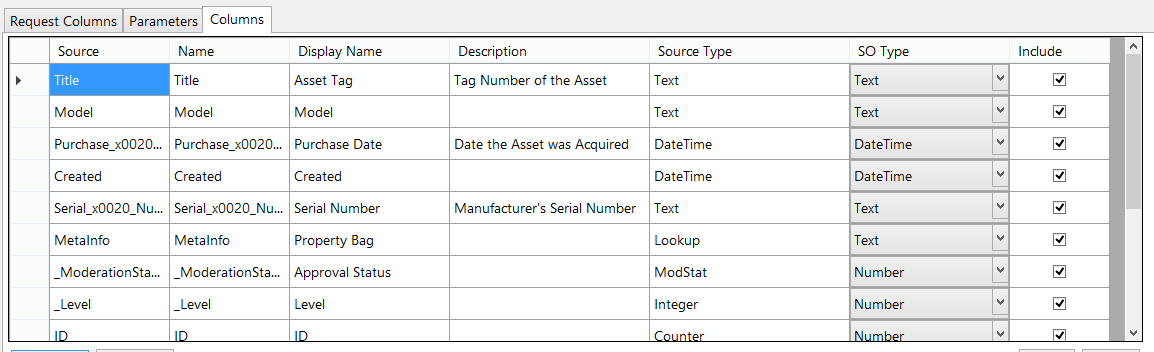
#### CAML Queries

CAML queries can be defined either by manually entering CAML or pressing the “Gen. CAML” button; which will use the entered Site URL and List URL to “generate” a default CAML query for that list. Note that the generate functionality is very rudimentary and is only provided as an aid. This tool is not a CAML designer so please use a proper authoring tool to create and test your CAML queries.

Once you have entered your CAML query, then you can press the “Detect” button on the Columns tab and it will execute your query and automatically detect the output columns.



Columns will then be displayed in the Columns tab.



|  |  |
| --- | --- |
| Name | Description |
| Source | The name of the column from the SharePoint, this much match exactly what is returned by SharePoint and (in the case of CAML) is the internal column name. For Search queries this is the name of the Managed Property. |
| Name | The column name that will be returned to K2. This is typically hidden in K2, but must be unique. |
| Display Name | The name of the column as it will be displayed in the K2 designers |
| Description | Long description of this column for the K2 metadata |
| Source Type | Column type are returned by SharePoint |
| SO Type | The desired output type to K2 |
| Include | Boolean valid indicating if this value will be returned and visible to K2. This can be used to exclude some columns that are, by default, returned from SharePoint CAML and Search Queries. |

For CAML queries the Detect feature populates the Display Name & Description from the metadata returned from SharePoint for the specific list.

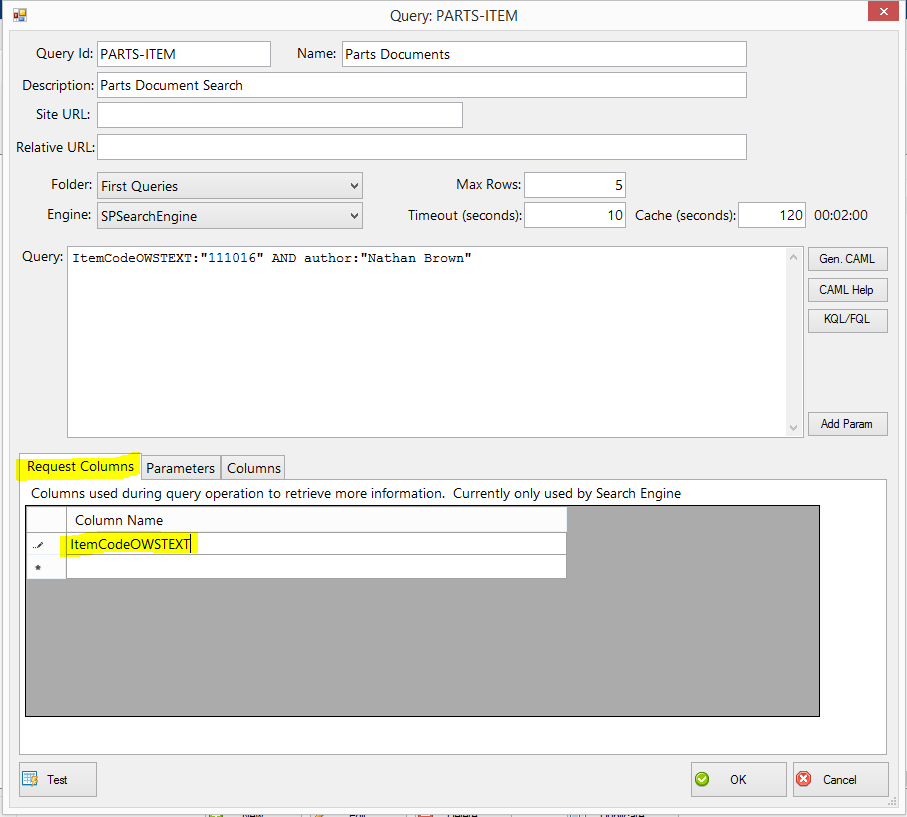
For clarity all dates are returned in UTC. There is an open enhancement to add a setting to allow this to be changed.

#### Search Queries

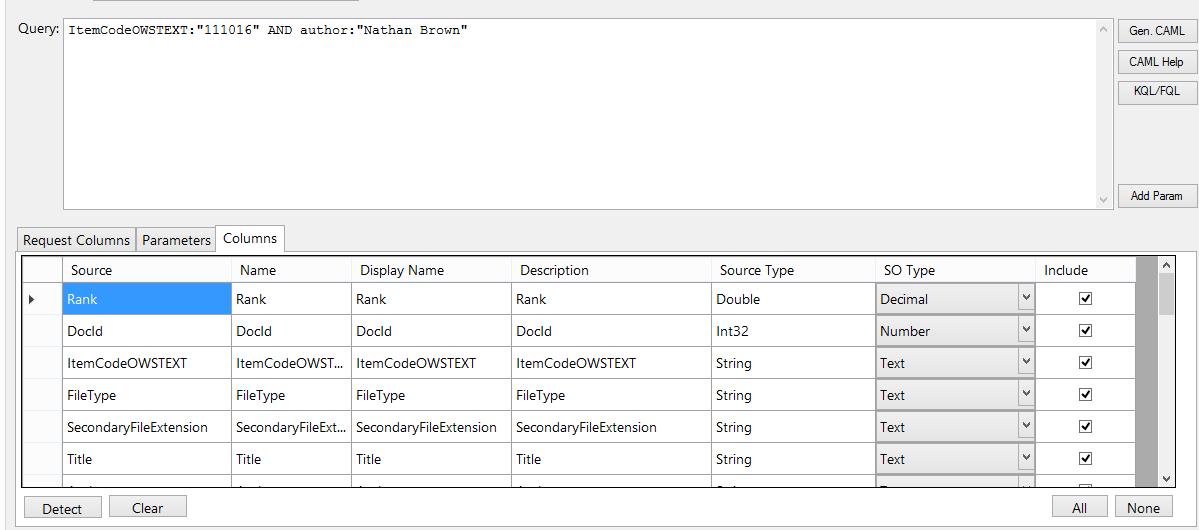
Please read the section on CAML Queries as it covers the processes to detect columns. There is no particular difference in output column configuration between the two methods.

Search queries are specified using KQL or FQL into the Query text box. The URL in most cases is the Root URL of your SharePoint site, but can be a sub-site to limit the scope of your Search result to that site. This should performs just as if you were searching that site.

Because Search queries do not specify return column in their query text, there is an addition tab “Request Columns” that is only used for Search Queries to request additional columns be returned with the Search Results. Add columns to this tab, then re-run the Detect function to ensure that they are returned with the Search. As with the Query Text column names must be Managed Properties.



Then we can use the same Detect feature as the CAML query and it will populate our columns.

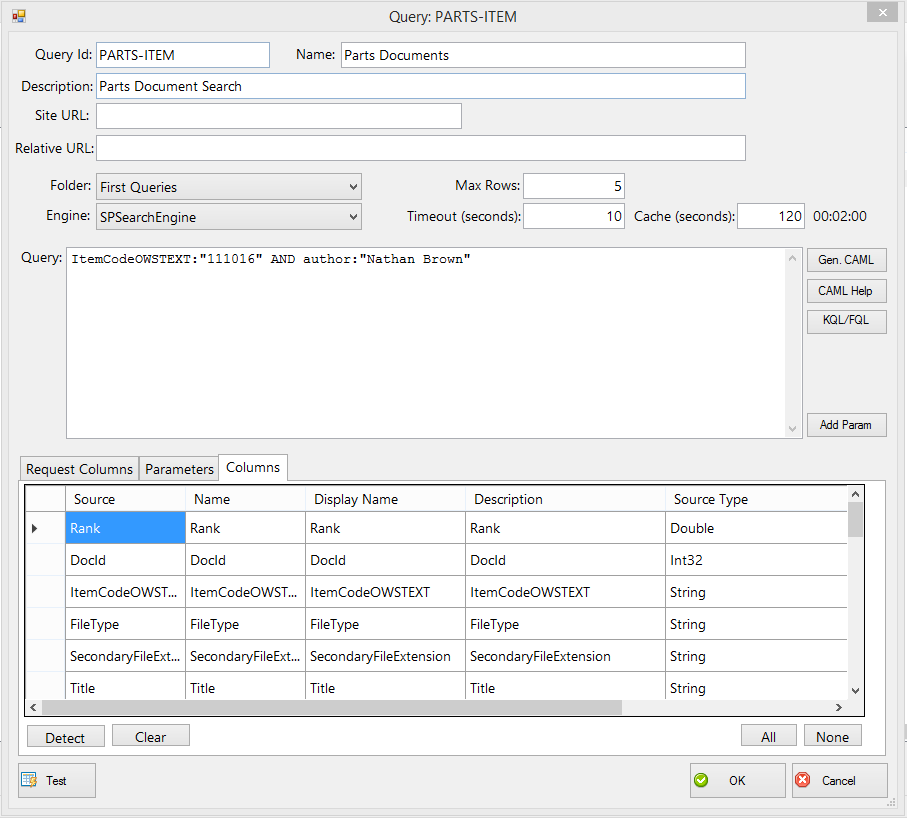


Notice that our column “ItemCodeOWSTEXT” is returned as part of the results set. This can be used with any Managed Property that is properly configured in SharePoint Search. Please see [this page](https://technet.microsoft.com/en-us/library/jj219630.aspx) from Mircosoft regarding more details about Managed Properties.

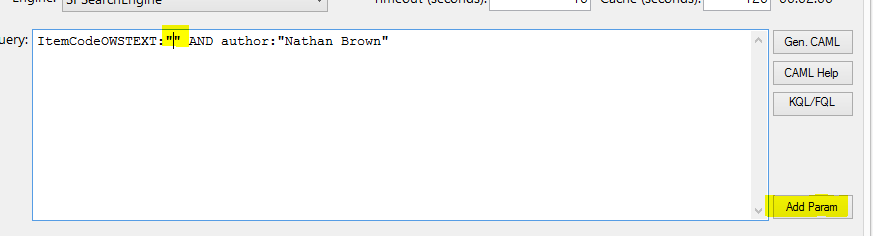
#### Parameters

Parameters are used to vary the query text, or URL of a query. They operate by inserting the value of a passed parameter into a location within the query text (for both CAML & Search) to make the query return different results.

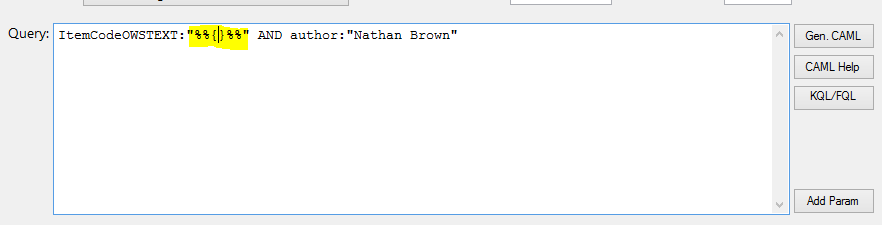
It is suggested to start with a fully functional query (CAML or Search) with hard coded values, then add parameters once that has been proven to work. So below we have a working Search Query with no parameters:



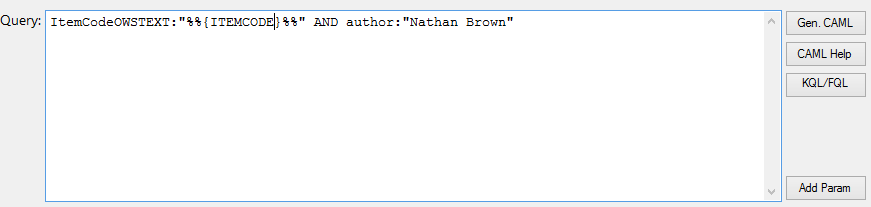
This query may be useful, but does not allow us to search for different results as it has no inputs. A logical input for this example is to use a parameter for the ItemCodeOWSTEXT field. So first remove the current hard-coded value, place the cursor at that location, then press the “Add Param” button.



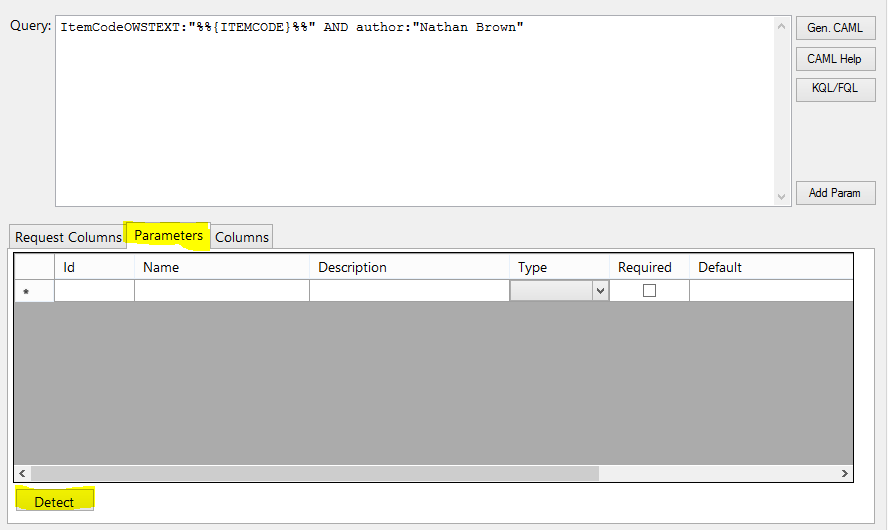
That will create the Parameter Macro syntax at that location:



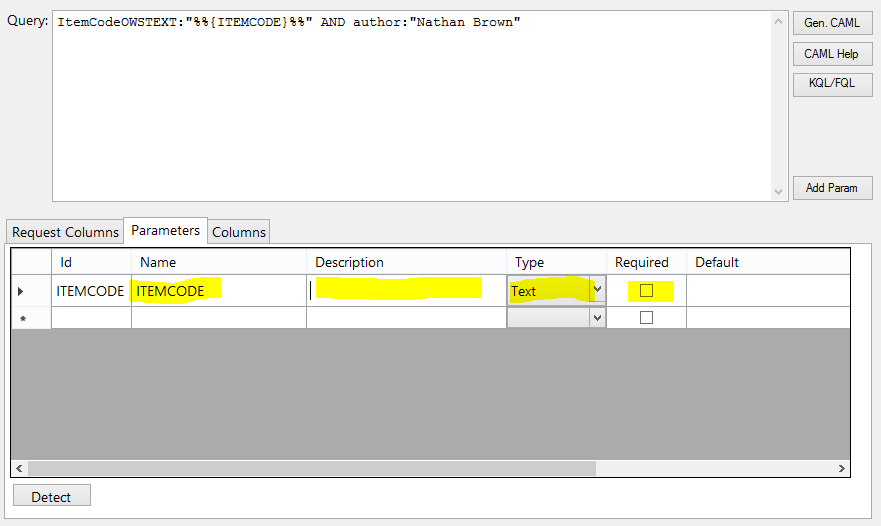
Give this parameter a name by typing a value:



Then navigate to the Parameters tab and press the Detect button:



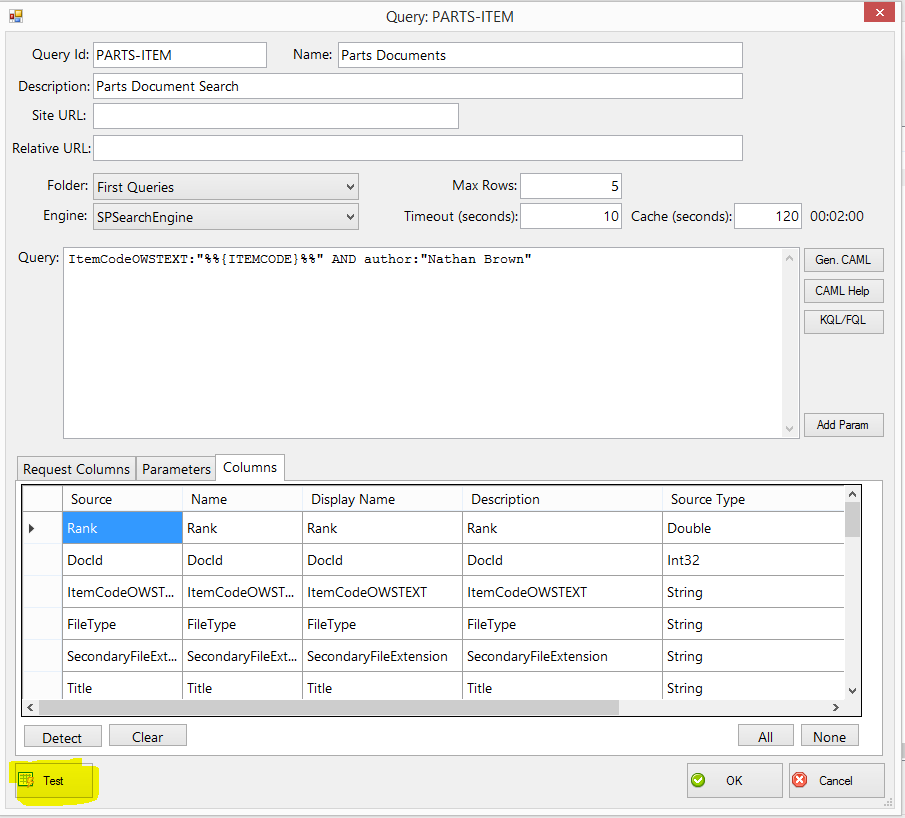
This will examine the query text for any macros with the proper syntax and add then as parameters to the query.



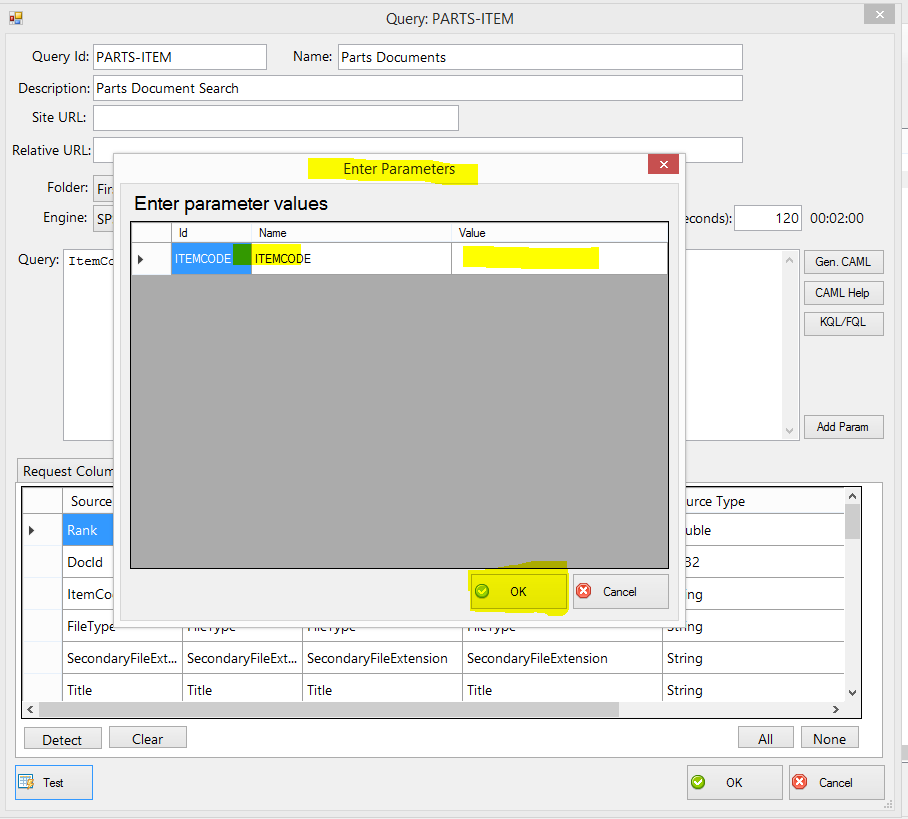
You must fill in the Description, Type, and Required properties of this parameter for it to function. Once you have created this parameter it will then be created as a Method Parameter for the service object.

#### Testing Queries

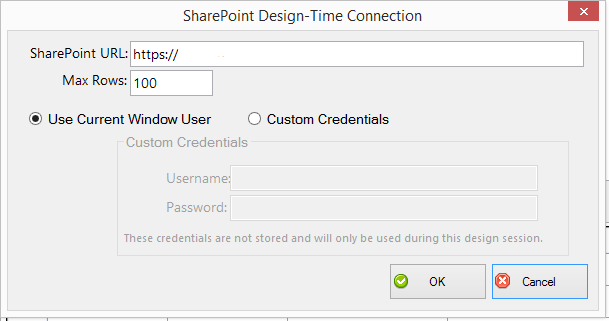
Once you have created a query it is best to do thorough testing to ensure that it will return the expected results. Pres the “Test” button on a completed query to execute the quest and display the results.



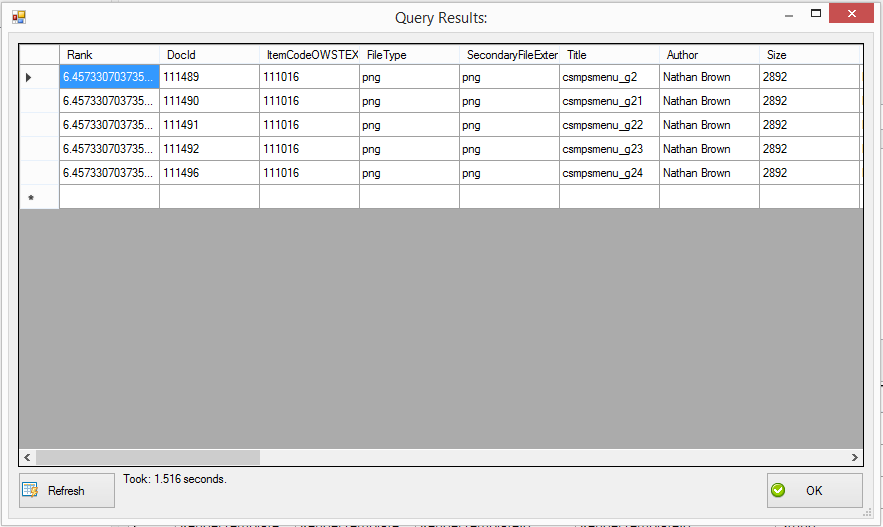
If your query has any parameters, you will be prompted to enter the values. This screen will display each time you test a quest so that you can test it with different values. The last value used will be saved for convenience.



If you have not already done so, you may be prompted for your design-time connection information.

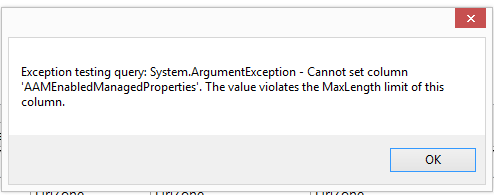


Lastly you will be presented with the results of your quest. These are exactly what you should expect when running this quest from a Smart Object. Note that this also includes the amount of time this query took in the lower left, along with a refresh button so you can run this multiple times to understand performance.

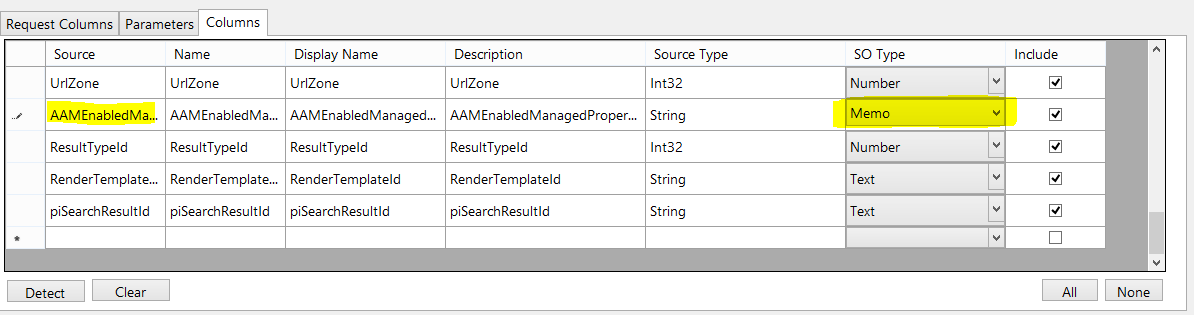


#### Tips

During design you may encounter this message:



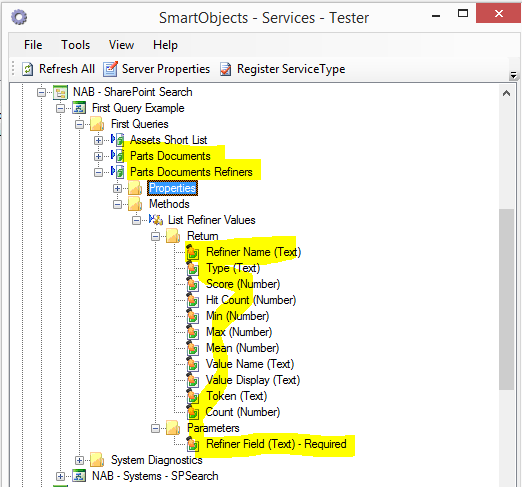
This indicates that a return field from your quest is too large for to fit into the field type specified. This is common for large columns that may default to be mapped into a Text field. Simply change the SO Type of the column to a Memo. It is a good practice to know and understand the fields you are working with so that you select the most appropriate SO Type for the return columns.



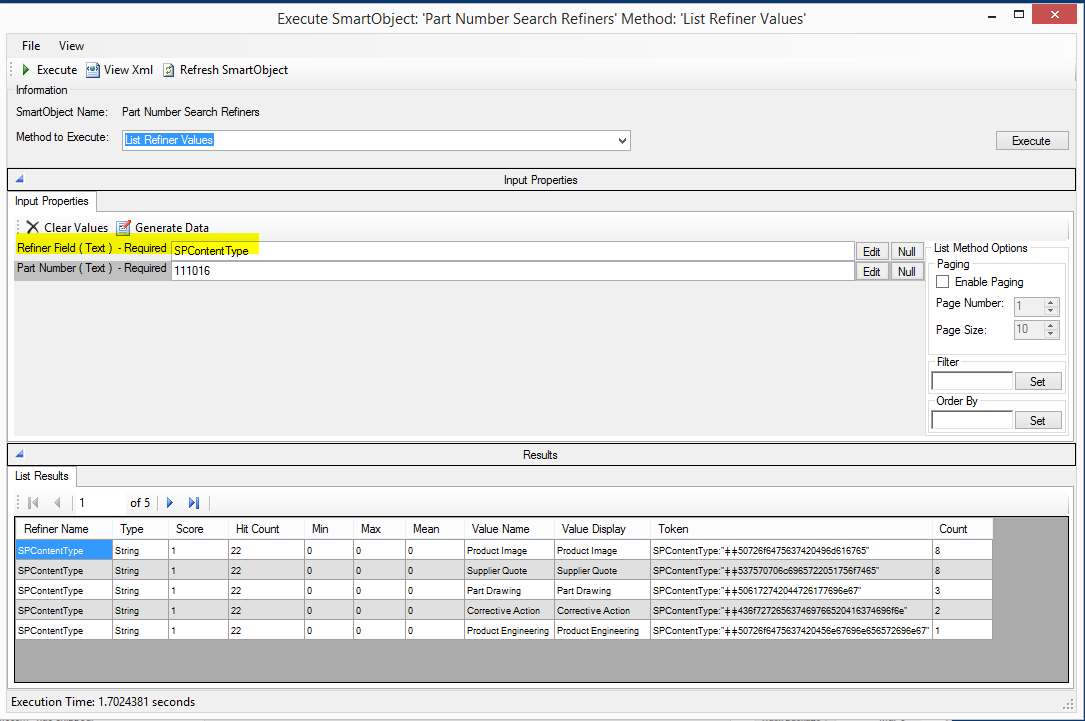
# Building Smart Objects

With the Queries in place and a Service Instance configured to read a configuration file, it is now time to build SmartObjects. This broker is designed to behave properly with K2 and thus building SmartObject should be identical to building them for any other system.

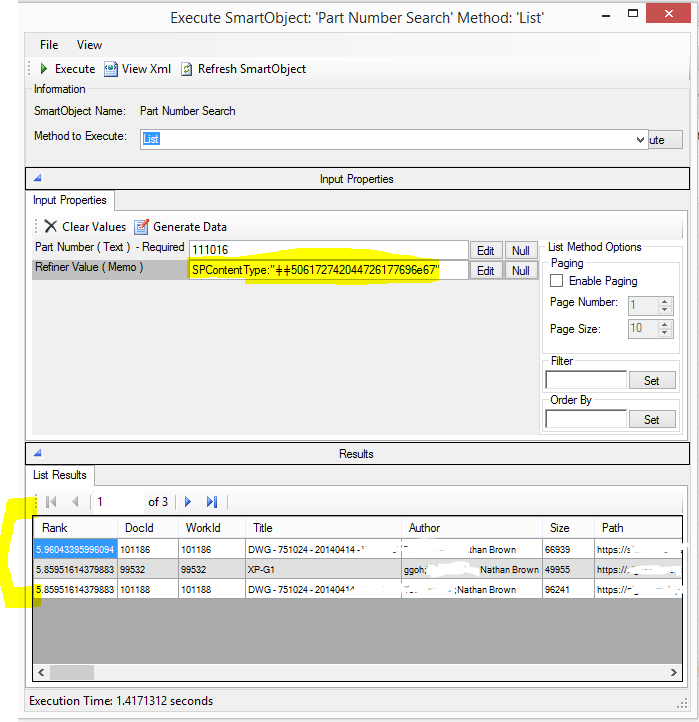
Smart Object that are created for CAML queries work as one might expect. The input parameters and output properties are exactly as are specified by the query. Search Queries, however, are a little different. You will notice that for each Search Query, two service objects are actually created. This is to support Refiners. Refiners are Managed Properties that can be used to further filter the result of a search by one or more fields.



The Refiners service object can be mapped to a Smart Object to return refinement values for a given query. The refinement will have the same parameter as the Search Query plus an addition parameter for the name of the Refinement field to be used. Here is an example output from a SMO for a refinement query:

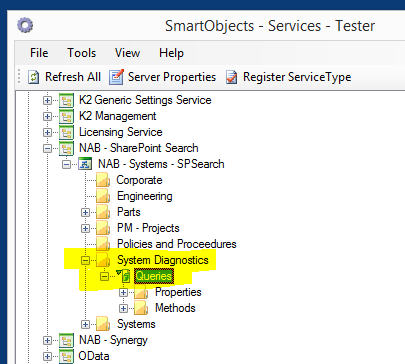


In this example we are Refining on the “SPContentType” managed property. The results indicate that they are 5 different Content Types that will be in the query results and the data indicates the Count of the expected results. We can then take the Token field and use that to refine our search. Notice we now run the standard Search query using the same parameters and copy the Token field.

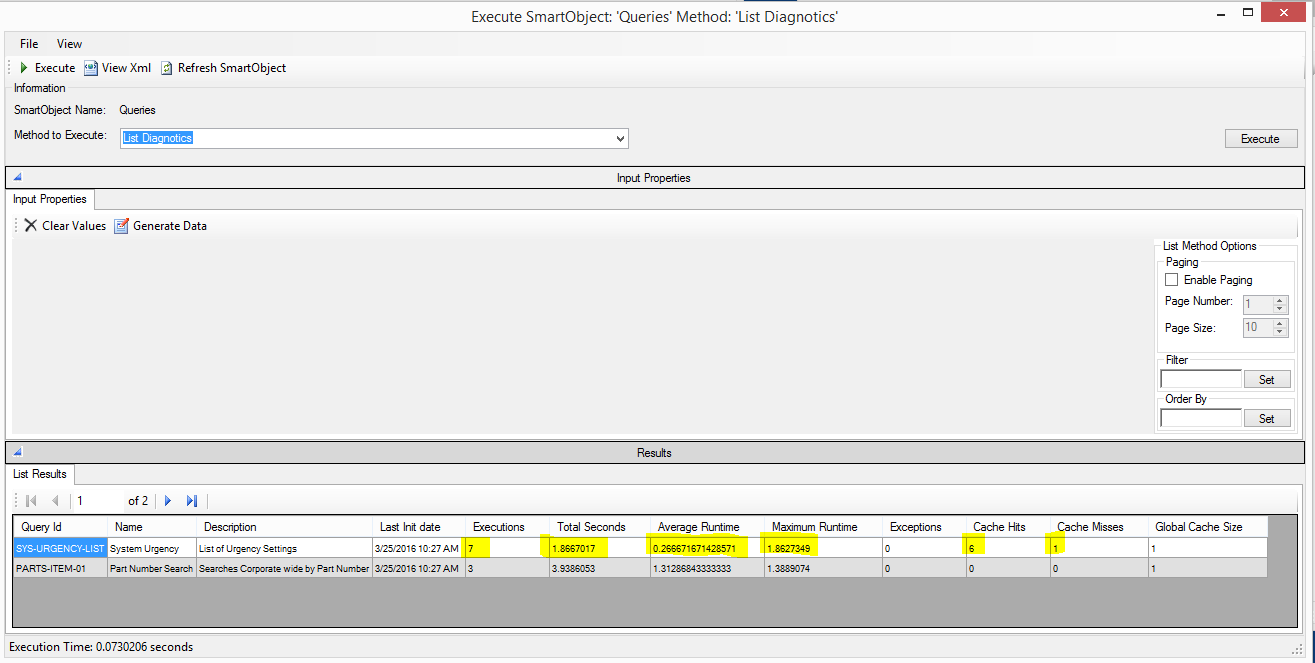


As expected we now receive 3 results which are all of the Content Type Part Drawing that we received from the Refiner query.

The second exception is for a diagnostic service object. You may have noticed that there is an additional Service Object exposed by this broker that is not controlled or defined by a query.



The System Diagnostics folder will contain any diagnostic service objects that can be used at runtime to check on the status of this Service Instance. Specifically the Queries object provides runtime performance counters for the queries in this instance so they can be checking at runtime. Creating a SmartObject for this will create a single method “List Diagnostics” (there methods may be added in the future).



As we see from the results of this method we see the date the broker was last initialized and statistics about each of the queries that have been executed; only queries that have executed will be displayed. We see a typical example of a CAML query at the top where the Maximum Runtime was 1.86 seconds for the first query with a single Cache Miss (for the first run), then the remaining 6 executions were Cache Hits bring down the average runtime of that query to .266 seconds! In this can each subsequent run only takes 0.019 seconds?

It is encouraged to build Smart Object for this diagnostic service object and use it to build a report showing the performance of your queries so that data can be used as the basis for further optimization.