tag.png





Advanced ADO Service Broker

V1.0

11/2/2018

*by Migael Moolman*

Contents

[Changelog 2](#_Toc529170664)

[Introduction 3](#_Toc529170665)

[Installing the Service Broker 3](#_Toc529170666)

[Files 3](#_Toc529170667)

[Installing and configuring 4](#_Toc529170668)

[Using the Service Objects 7](#_Toc529170669)

# Changelog

#### v1.0

Initial release

# Introduction

This service broker is based on the ADO Query Broker originally developed by Johnny Fang. Credit and Thanks goes to Johnny for providing the code and writing the baseline code that this broker is based on.

There were some shortcomings in the original broker which are addressed in this Service Broker. There are also some new capabilities.

* The original service broker required that the correct Method be used based on the exact number of columns that were expected to be returned by the input ADO Query. This broker handles that limitation by providing a single Query method that can return 0-15 columns. (Unused columns will be returned as empty and if more than 15 columns are returned, only the first 15 will be used so no more errors when there’s a mismatch)
* The original service only allowed a single input, the ADO Query String. Even though you could dynamically build up the query string in a Workflow/SmartForm, it was not possible to build up that string within a Smart Object directly which limited some of the possibilities. This broker provides the ability to insert placeholders (“{value1}” to ”{value15}”) in the ADO Query String and also provides additional inputs (Values 0-15) which are inserted into the placeholders at runtime.
* Additional convenience Method such as getting First, Last and Item at Index have also been provided.
* There is also the ability to retrieve row data as a Joined string for scenarios where row data needs to be transformed into a joined string. Splitting is also available to reverse the above process when needed.
* More coming soon.

# Installing the Service Broker

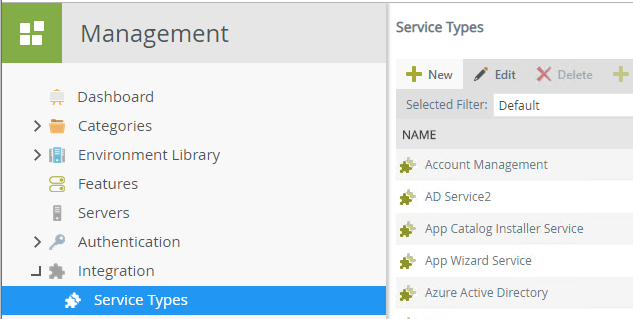
## Files

Within the Service Broker package, you will find the following folder structure

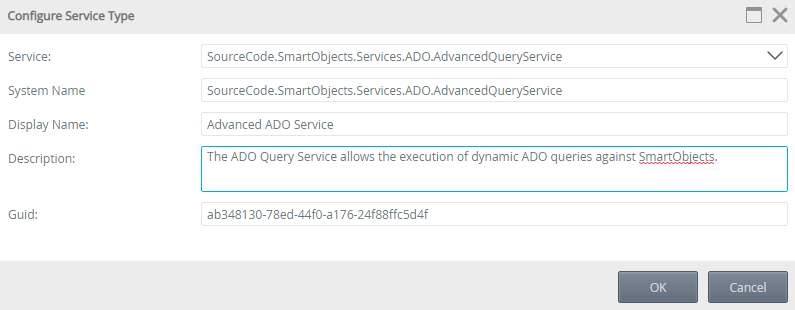
* **Advanced ADO Service Broker** – this folder contains a compiles version of the service broker files that are ready for use
  + **SourceCode.SmartObjects.Services.AdvancedADOService.dll** – This is the ADO Service Broker
* **Advanced ADO Service Broker Code** – this folder contains the source code for the service broker

## Installing and configuring

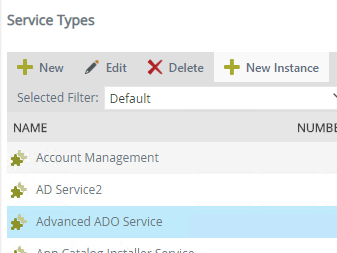
1. Copy all the files from the “Advanced ADO Service Broker” folder mentioned above into the “K2 blackpearl\Service Broker” folder on every K2 server within an environment. This folder is typically “C:\Program Files\K2\ServiceBroker”
2. Open K2 Management and navigate to Integration->Service Types
3. Click on “New” button at the top to register a new Service Type.



1. Select the Service DLL from the dropdown list and specify a Display Name and Description and click “Add”



1. Select the Service Type you just created and click on “New Instance”

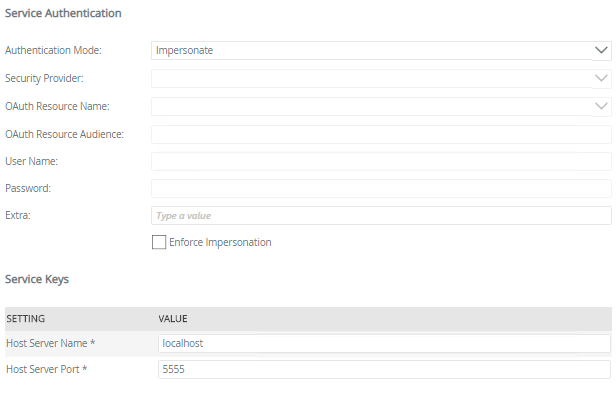


1. Define the Display Name and Description for the Service Instance.

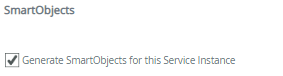


1. Configure the Authentication Mode (Default is “Impersonate”) and specify each of the following parameters as required:

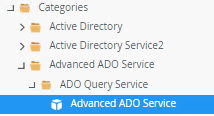
* **Host Server Name** – This is the Server Name/DNS Name or IP Address of the K2 server the service broker should connect to (default is LOCALHOST). Do not change this unless you want to query against another K2 Server.
* **Host Server Port** – This is the Port used by the K2 server that the service broker should connect to (default is 5555). Do not change this unless your K2 Server uses a not default port.

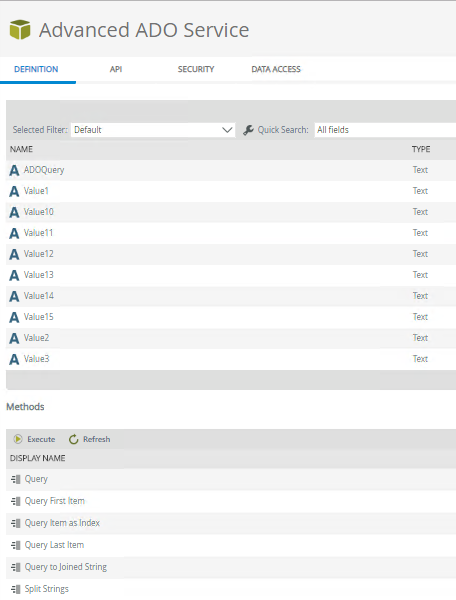


1. Check the Checkbox for “Generate SmartObjects for this Service Instance” and click



1. Once registered you have the following Service Objects and Methods to work with:





# Using the Service Objects

Here are some examples of what can be done using this service broker:

* Retrieve a Filtered Distinct list of values from any SMO
* Count total number of Items from any SMO list method
* Dynamically perform Left, Right, Inner, Outer or Cross Joins on SMO data.
* Create Generic SmartObjects. E.g. you can put placeholders in the ADO query which are populated at runtime by values passed in as input values.

This Service is very useful for scenarios where you need more flexibility in joining SMO data and querying across multiple SMOs from different sources not just SQL.