

**Data Import Service Broker Overview**

TO ALLOW BULK UPLOADING OF EXCEL / CSV DATA TO SMARTOBJECTS

17 August 2022

Table of Contents

[Introduction 3](#_Toc84934465)

[Methods and Property descriptions 4](#_Toc84934466)

[Methods 4](#_Toc84934467)

[Properties 5](#_Toc84934468)

[Considerations 7](#_Toc84934469)

[Bulk Import 7](#_Toc84934470)

[Non-bulk Import 7](#_Toc84934471)

[Import performance Comparison 7](#_Toc84934472)

[Testing 8](#_Toc84934473)

[Testing the CSV import 11](#_Toc84934474)

[Testing the Excel import 13](#_Toc84934475)

[Installation 15](#_Toc84934476)

|  |  |  |
| --- | --- | --- |
| Version | Author | Notes |
| 1.0 | Johnny Fang | Initial Release. |
| 1.0.1 | Johnny Fang | Fixed issue in handling spaces in column names. |
| 1.0.2 | Justin Warwick | Added support for empty cells.  Added support to specify to remove space in column names (instead of replacing with underscore). |
| 1.0.3 | Roy Higgs  Johnny Fang | Added support for Excel DateTime values.  Known Issue: Date and Time values not supported yet due to a bug in the bulk insert API. Ticket #105510 |
| 1.0.4 | Paul Kelly | Added support for CSV files.  Added non-bulk upload methods.  Added named worksheet selection for Excel files.  Added duplicate column handling for Excel files.  Added row header index to allow you to specify where the data in the Excel file starts.  Skip import of empty Excel rows.  Removed the A-ZZ column limitation – at the cost of allowing empty columns. |
| 1.0.5 | Paul Kelly | Added failed error row reporting to the non-bulk upload methods |
| 1.0.6 | Paul Kelly | Fix for empty cell values causing misalignment of column data.  https://github.com/paulk-k2dev/DataImporterServiceBroker/issues/1 |

# Introduction

This is a service broker which allows you to upload table data from an Excel or CSV file to a SmartObject.

This uses the OpenXML SDK for Office when doing the Excel importing and utilises the samples from [MSDN](http://msdn.microsoft.com/en-us/library/office/gg575571(v=office.15).aspx).

The [GenericParser](https://github.com/AndrewRissing/GenericParsing) component was used for CSV importing.

This was tested on K2 blackpearl 4.7 and the solution was built using Visual Studio 2015.

Note that this service broker was created as a proof of concept and is provided as-is. It has been tested to verify functionality but has not gone through a full QA cycle or tested under heavy load.

It is based upon the Excel Service broker <https://community.k2.com/t5/K2-blackpearl/Excel-Import-Service-Broker/ba-p/65814> and has added a number of new features added based upon the comments in the discussion thread.

# Methods and Property descriptions

The data service provides 8 methods that can be used depending on the data being imported and the types of column data that they contain.

## Methods

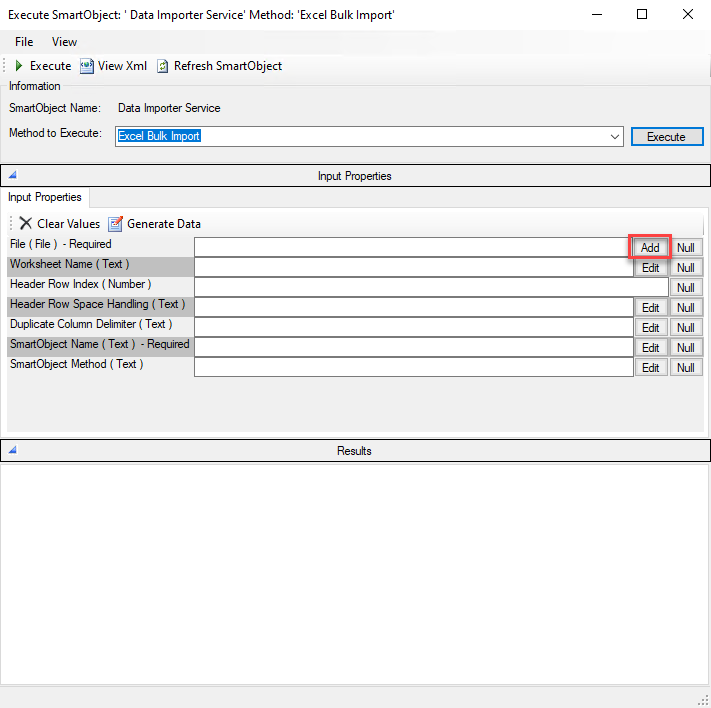
|  |  |  |
| --- | --- | --- |
| Method Name | Description | Properties |
| Excel Bulk Import | Imports rows from an Excel file by inserting all rows in one go. | File – **Required** Worksheet Name Header Row Index Header Row Space Handling Duplicate Column Delimiter SmartObject Name – **Required** SmartObject Method |
| Excel Bulk Import With Transaction | Imports rows from an Excel file by inserting all rows in one go and appends a transaction column and value to the imported data. | File – **Required** Worksheet Name Header Row Index Header Row Space Handling Duplicate Column Delimiter SmartObject Name – **Required** SmartObject Method Transaction Id Property Name – **Required** Transaction Id Value - **Required** |
| Excel Import | Imports rows from an Excel file by inserting all rows one by one. | File – **Required** Worksheet Name Header Row Index Header Row Space Handling Duplicate Column Delimiter SmartObject Name – **Required** SmartObject Method |
| Excel Import With Transaction | Imports rows from an Excel file by inserting all rows one by one and appends a transaction column and value to the imported data. | File – **Required** Worksheet Name Header Row Index Header Row Space Handling Duplicate Column Delimiter SmartObject Name – **Required** SmartObject Method Transaction Id Property Name – **Required** Transaction Id Value - **Required** |
| Csv Bulk Import | Imports rows from a Csv file by inserting all rows in one go. | File – **Required** Header Row Space Handling Column Delimiter Text Qualifier SmartObject Name – **Required** SmartObject Method |
| Csv Bulk Import With Transaction | Imports rows from a Csv file by inserting all rows in one go and appends a transaction column and value to the imported data. | File – **Required** Column Delimiter Text Qualifier SmartObject Name – **Required** SmartObject Method Transaction Id Property Name – **Required** Transaction Id Value – **Required** |
| Csv Import | Imports rows from a Csv file by inserting all rows one by one. | File – **Required** Header Row Space Handling Column Delimiter Text Qualifier SmartObject Name – **Required** SmartObject Method |
| Csv Import With Transaction | Imports rows from a Csv file by inserting all rows one by one and appends a transaction column and value to the imported data. | File – **Required** Column Delimiter Text Qualifier SmartObject Name – **Required** SmartObject Method Transaction Id Property Name – **Required** Transaction Id Value – **Required** |

## Properties

The following properties belong to the SmartObject

|  |  |
| --- | --- |
| Property Name | Description |
| File | The file to be processed **\*** |
| Worksheet Name | The name of the worksheet to parse in an Excel file. Defaults to the first worksheet if no name is specified or the Worksheet Name provided is not found. |
| Header Row Index | The index of the row that contains the headers in an Excel file. Defaults to 1 |
| Header Row Space Handling | How to handle spaces in the column names in the source data.  Remove – Remove will remove spaces in the Column Name  Replace – Replace will replace spaces in the Column Name with an underscore \_  Defaults to Remove  SmartObject properties system names do not allow spaces so do the above action to make them match. |
| Duplicate Column Delimiter | If your source Excel file has multiple columns named the same then the imported data will be appended into 1 destination column. The values will be delimited with the specified character. Defaults to ; |
| Column Delimiter | The character that indicates the column delimiter in a Csv file. Defaults to , |
| Text Qualifier | The character that indicates the text delimiter that surrounds a value in a Csv file. Defaults to ". |
| SmartObject Name | The name of the SmartObject that the data should be imported into. |
| SmartObject Method | The method on the SmartObject that should be called to save the data. Defaults to Create. |
| Transaction Id Property Name | The name of the property on the SmartObject that indicates the transaction. To be used to identify the data on an import. |
| Transaction Id Value | The value of the property that should be set against each imported row on the SmartObject. |

**\*** whensetting the File Property in the SmartObject tester you will need to click Add to attach the file rather than typing in the path to the file.



# Considerations

## Bulk Import

The bulk methods will place less load on the K2 server as the saving of all the rows gets performed at once. You should prefer this method for large data sets.

***There is a known problem in the SmartObject Execute Bulk method which means that Date / Time values may not be saved against the SmartObject. If your data has Date / Time values then test thoroughly or use the equivalent non-bulk method. A warning message will be added to the SmartObject execution result message.***

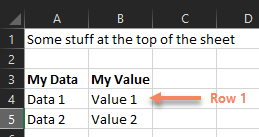
## Non-bulk Import

The non-bulk equivalents will import the data row by row. This will place more load on the K2 server as it will process and call the Create method on the SmartObject for each row in the source data.

**If there is a problem with any particular row then the importing of that row will be skipped and the rest of the data will be imported, The Upload Status will be Partial. If any rows that fail to import there will be the following content in the resulting message:**

The following rows failed to upload: 1, 4, 999.

You can use this ‘The following rows failed to upload:’ as a key to split the string out if you need to do anything else with those failed rows. Please note that the failed row value in the case of Excel files with skipped header rows refers to the ordinal position of that row and not the Excel displayed row.



Use the non-bulk methods if your SmartObject contains date / time fields and data you want to import data to that column or if the number of rows being imported is low.

### Import performance Comparison

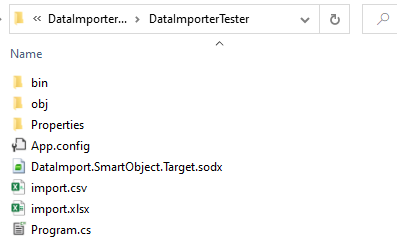
Using the included Excel file, with the example data duplicated to 2000 rows instead of 2, the following import metrics were achieved on a small to mid-sized K2 server. As is always the case, test to see how it performs for you set-up / requirements as your mileage may vary.

|  |  |
| --- | --- |
| Method | Average Duration (seconds) from 10 runs |
| Excel Bulk Import | 1.652 |
| Excel Import | 11.235 |

# Testing

A sample spreadsheet, Csv and destination SmartObject can be found as part of the zip file.

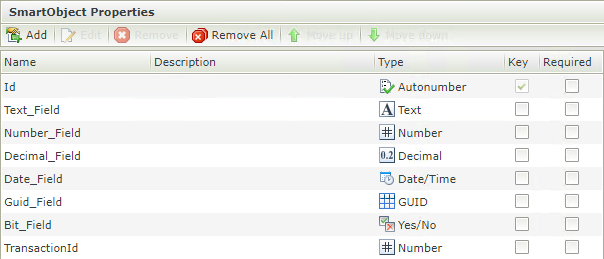
These files are in the DataImporterTester project folder:



Import the SmartObject to your K2 server via the SmartObject tester and you can then use it to import the data from the sample files.

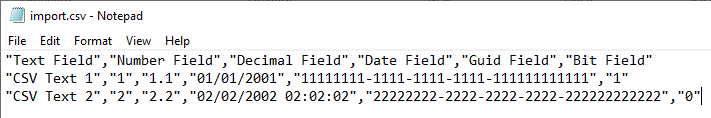
|  |  |  |
| --- | --- | --- |
| S/N | Step | Screenshot |
| 1 | Open the SmartObjects Services Tester and choose Import SmartObjects |  |
| 2 | Open the DataImporterTester folder.  Switch the Type to SmartObject File  Select the DataImport.SmartObject.Target and click **Open** |  |
| 3 | Click **OK** |  |
| 4 | Click **Edit** |  |
| 5 | Enter *Data Import Service\Data Import Service* for the Deployment Category  Click **Exit** |  |
| 6 | The SmartObject will now be available. |  |

You will see that the column names in both the Excel and CSV files have a space and the equivalent SmartObject properties are named with an underscore. Therefore the “**Header Row Space Handling**” property should be set to “**Replace**” when testing.



## Testing the CSV import

The following example CSV file has been included for testing.

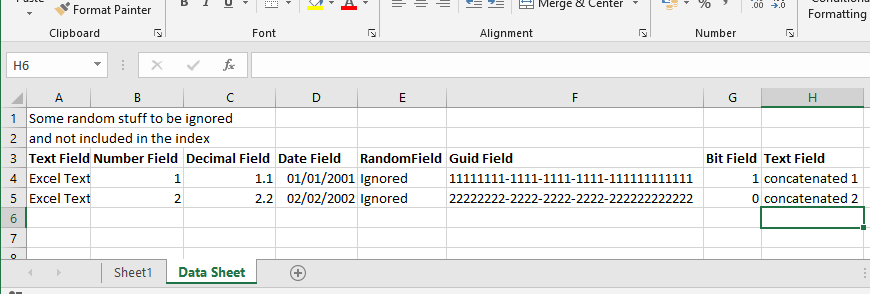


*Please note that although the values in the CSV are text delimited with " your data does not need to be and is done to show the Column Delimiter property and how to use it in the below example.*

|  |  |  |
| --- | --- | --- |
| S/N | Step | Screenshot |
| 1 | Select the *Csv Bulk Import With Transaction* method on the *Data Importer Service SmartObject*  Click **Add** and choose the *import.csv* file  Enter *Replace* for Header Row Space Handling  Enter " for the Text Qualifier  Enter *DataImport.SmartObject.Target* for the SmartObject Name  Enter *TransactionId* for the Transaction Id Property Name  Enter *1000* for the Transaction Id Value  Click **Execute**  **Note that the defaults for Column Delimiter (*,*) and SmartObject Method (*Create*) will be used.** |  |
| 2 | Select the *Get List* method on the *DataImport.SmartObject.Target* SmartObject.  Click **Execute**  You will see the data has been imported to the relevant columns and that the value of 1000 has been added to each imported row to the TransactionId column. |  |

## Testing the Excel import

The following example Excel file has been included for testing:



Please note that there is a column RandomField that will be ignored on import.

There are two columns called TextField that will have their values concatenated into the Text\_Field property of the SmartObject.

The Worksheet to import “Data Sheet” is not the first worksheet in the spreadsheet.

|  |  |  |
| --- | --- | --- |
| S/N | Step | Screenshot |
| 1 | Select the *Excel Import* method on the *Data Importer Service SmartObject*  Click **Add** and choose the *import.xlsx* file  Enter *Data Sheet* for the Worksheet Name  Enter *3* for the Header Row Index  Enter *Replace* for Header Row Space Handling  Enter *DataImport.SmartObject.Target* for the SmartObject Name  Click **Execute**  **Note that the defaults for Duplicate Column Delimiter (;) and SmartObject Method (*Create*) will be used.** |  |
| 2 | Select the *Get List* method on the *DataImport.SmartObject.Target* SmartObject.  Click **Execute**  You will see the data has been imported to the relevant columns, the RandomField has been ignored and that the Text\_Field values have been concatenated. |  |

# Installation

This section details the installation of the service broker.

|  |  |  |
| --- | --- | --- |
| S/N | Step | Screenshot |
| 1 | Unzip the project and locate the *DataImporter.SmartObjects.Services.Broker.dll, DocumentFormat.OpenXml.dll* and *GenericParsing.dll* files from the *.zip* file and copy them to ***[Installed Dir]*\ServiceBroker**  **If K2 is running as a farm then do this for all servers.** |  |
| 2 | Open the SmartObjects Services Tester and click on Register ServiceType |  |
| 3 | Select *DataImporter.DataImporter.ServiceBroker* |  |
| 4 | Enter *Data Import Service* for both the Display Name and Description Fields. |  |
| 5 | Click **Add** |  |
| 6 | Right click on the newly created Service Type and select **Register ServiceInstance**. |  |
| 7 | Specify the *Authentication Mode* and the *K2 Server* and *Port* to the K2 host server.  In this case, *ServiceAccount* mode is used and we have left the default settings of **localhost** for the K2 Server and **5555** for the Port. |  |
| 8 | Click **Next** |  |
| 9 | Click **Add** |  |
| 10 | Generate the default SmartObjects. Right click on the service instance and select **Generate SmartObjects**. |  |
| 11 | Click **OK** |  |