Data migration tools

Contents

1 Import.exe

The Adlib tool *import.exe* was introduced originally to offer a faster way of importing than DBSETUP, the old database management tool. DBSETUP has been replaced by Adlib Designer, in which importing is just as fast as with import.exe. So which program you want to use for importing is a matter of personal preference.
Import.exe (6.0 and higher) recognizes DOS, ANSI, UTF-8 and Unicode (big endian as well as little endian) exchange files and imports the data from them, regardless of the type of the database. Then for ANSI and DOS databases the following applies: if there are non-importable Unicode characters in the exchange file, they will be replaced by "?" (a question mark). Non-UTF-8 exchange files will be interpreted as being ANSI, except when the database type is DOS.

Create a backup

Importing data into existing databases can be a far-reaching procedure. Therefore you should create a backup of all your databases before you start importing, just to be safe. That way, you can always repair any errors. See the Installation guide for Museum, Library and Archive for more information about creating backups.

You have to define your import job in Designer, but you can execute it with import.exe. The only things you need are this executable and an Adlib import job (.imp file).

You start import.exe command line in a command line window. The syntax is as follows:

```
IMPORT [-?][-a:<adapl>][-i:<jobname>]
[-r:<recovery file>][-s][-u:<username>]
```

Everything in between square brackets [ ] is optional, and everything in between sharp brackets <> should be replaced by an actual name. And you should leave all brackets out.

The following, for example:

```
IMPORT -i:..\data\MyImportJob
```

would just execute the import job called `MyImportJob` in the directory `..\data`.

There are even more optional command line options though, which you can sum up in random order behind `IMPORT`. Below you'll find all possible options explained.

`-?`	`IMPORT -?` shows Help similar to this document.
`-a:<adapl>`	Specify an adapl to overrule any import adapl set in the import job definition.
`-b:<exchange file>`	Through this option you can execute an import job with another exchange file (*Source data file*) than specified in the import job. This way you can use the same import job to import multiple exchange files quickly after one another into a database. For example: `IMPORT –i:..\data\MyImportJob –b:externaldataset2` Note that you should never use the `–s` option whenever you are using `–b`, because the previous import will be (partially) overwritten otherwise.
`-c:<cross reference>`	Creates a cross reference. This option makes an adlib.lst file with therein a list of fields and tags of all databases in the concerning folder (the data folder). A list of all linked fields is created too.
`-d:<database> <index tag>`	Dump an index to file, like you did in DBSETUP with **`F9`**, for an index in a CBF database (SQL not supported). The dump is created in a text file with the name of the field and the *.dmp* extension, in the current folder. Start de command from the Adlib *\data* folder. Example of a call: `..\executables\import –d:thesau te` A special case is the *wordlist.idx* index: this index collects all unique words from all long text fields in all CBF database and assigns them a unique number. The free text indexes which index the long text fields directly, only point to the word

	number from the wordlist. To dump the wordlist to a file, use the following special syntax: `import -m -d:document wordlist.idx` Point to a database that has free text indexes, like `document` or `collect`. In an Adlib SQL database you can inspect all indexes directly via SQL Server Management Studio, so there is no necessity to dump indexes to a file via import.exe.
`-exclusive`	Exclusive access. This option specifies that the import executable has exclusive access to the database. Now, locking and similar operations cannot occur and the program code that checks for any occurring file and record locks will be skipped, making the import process much faster. The option can only be used when executing an import job (`-i:job name`), and for re-indexing (`-x`).
`-fix`	Fix CBF file while reindexing primary index. Only use `-fix` behind `-x:database %0`. If there are more records with the same priref in a database (which almost never happens, by the way), then `-fix` indexes the most recent record of these. The older version(s) of the record are moved to a file named *records.log*. This enables the database manager to check the removed records, and to import them again if necessary. (In *records.log*, records are stored in the Adlib tagged format.)
`-i:<job name>`	Opens the given import job and executes it.
`-l`	This option is obsolete.
`-o`	(This is the letter o, not the digit zero.) With this option you have all records to be imported registered in the logging file that must have been set for the database into which you import.

	Note that normally, importing is not being logged. Only for *import.exe* you can switch logging on explicitly; in Designer this is not possible yet.
-q:<database name>	With this option you'll empty the database indicated behind `-q:`, before the rest of the import job is executed. All contents will be removed! A database name equals the name of the related *.inf* file, but without the extension. If the relevant *.inf* file is located in a folder different from import.exe, then precede the database name by the path to the file. Example: `import -q:..\data\collect`
-r:<recovery file>	Start recovery through the specified recovery file. See the Designer Help and the *Installing Museum, Library and Archive* guide, for information about recovery files.
-s	Force auto record numbering; the first record starts with number 1. Do not use –s in combination with –b.
-u:<username>	Force username to be used. With this you can execute an import job as someone else. This is handy for example when you are logged on as administrator on a server at which you run an import job, then you can execute it under your own name with the option `-u:erik` (fill in your own name of course). Or the option can be used to mark these records as created during an import job. Therefore use: `-u:import`. In the records the *Input name* now becomes *import* instead of your own name.
-v	With the command-line `-v` option (this was called `-f`, prior to 6.1) you indicate that during import, *Use/Used for* relations must be ignored. This is only relevant when you import a thesaurus file (or another file with such relations), import it

	with the marked import job option *Process links*, and when in the target file all internal links are on link reference. This is because *Process links* not only automatically replaces non-preferred terms by preferred terms in external links, but also in internal links on link reference. A non-preferred substitution is convenient for an import in a catalogue but not in a thesaurus, because in there both kinds of terms are actually defined. So if links *should* be processed, but not the *Use/Used for* relations, then use `-v` when executing import.exe.
`-x:<database> <index tag>`	Re-index a specific index for the database e.g.: `import -x:thesau te.` The argument `-x` re-indexes the index that is known as `<index-tag>` in the database `<databasename>`. If you also add the argument `-exclusive`, re-indexing is done very quickly. Execute this command of course in the *data* subfolder of your application. When you are re-indexing a unique index, and duplicates are found, then re-indexing will continue until the complete database has been processed; the advantage of this is that –x always creates a complete index. Reports on duplicate terms are placed in the .err-file. In an Adlib SQL database you can only re-index already existing indexes, while for Adlib CBF database this option can be used as well to create new indexes automatically.
Options **`-i`** and **`-r`** are mutually exclusive, so they cannot be used together.	

2 Export.exe

The Adlib tool *export.exe* was introduced originally to offer a faster way of exporting than DBSETUP, the old database management tool. DBSETUP has been replaced by Adlib Designer, in which exporting is just as fast as with export.exe. So which program you want to use for exporting is a matter of personal preference.
Export.exe (6.0 and higher) exports data from DOS, ANSI or UTF-8 encoded Unicode databases to exchange files in UTF-8.

But you have to specify your export job in Designer. So both programs need an .exp file to execute. Start export.exe from the command line in a command line window. The syntax is as follows:

```
EXPORT [-?][-e:<jobname>][-l][-s][-u:<username>]
```

Everything in between square brackets [ ] is optional, and everything in between sharp brackets <> should be replaced by an actual name. And you should leave all brackets out.

The following, for example:

```
EXPORT –e:..\data\MyExportJob
```

would just execute the export job called `MyExportJob` in `..\data`.

You can sum up the (optional) command line options in random order behind `EXPORT`. Below you'll find all possible options explained.

-?	`EXPORT -?` shows Help similar to this document.
-e:<job name>	Opens the given export job and executes it.
-l	Export including links.
-s	Force auto record numbering; the first record starts with number 1
-u:<username>	Force username to be used. With this you can execute an export job as someone else. This is handy for example when you are logged on as administrator on a server at which you run an export job, then you can execute it under your own name with the option `–u:erik` (fill in your own name of course).

3 Adcopy

3.1 Introduction

The main purpose of the command-line Adlib Adcopy tool is to copy and convert an Adlib SQL database in which record data is still stored in binary format (pre-SQL Server 2005) to an Adlib SQL database in which record data is stored in XML format. The XML format is a requirement for the Adlib API in order to work with the thesaurus search operators, such as the *generic* search operator.
During this conversion the tool copies and changes the .inf files of your database (even pre-6.4 data structures), it adjusts the SQL data tables to reflect the current (6.6.0) Adlib SQL database structure and then copies the data tables into another Adlib SQL database where it also rebuilds all indexes. The contents of pointer files are also copied along in this conversion.

The tool can also be used to:

- migrate data from an Adlib Oracle database to an Adlib SQL Server database. (Adcopy can read both binary and XML data columns, but will always write the result as XML columns.)
- migrate data from one server to another. Normally you would use a backup and restore to move data from one server to another, but by using Adcopy you make sure that only tables which are actually in use will be migrated. Since the Adcopy program also rebuilds all indexes you are sure that after the copying process all indexes are up-to-date. Note that Adcopy is much slower at this than using a backup/restore process: Adcopy typically does about 100 catalogue records per minute.
- place data "in the cloud". Adcopy can be used to upload Adlib data to a Microsoft Azure SQL server in the cloud.
- create a stripped (public) version of your data. The Adcopy program can remove fields from the data whilst copying. This can be used to remove sensitive information from an SQL database before it leaves the premises, for instance to be hosted by a third party.

This manual currently only describes the SQL binary to XML conversion type, and the procedure to create a (stripped) copy of your database. Because of some overlap, the next chapter explains how to achieve either goal in a single procedure. Basically this procedure comes down to the following:

1. Adcopy needs a target data folder other than the one in your production environment to copy altered .inf files (database structure files) to. If you just want to convert binary data in your SQL database to XML data, you can create a temporary, differently named data folder for this purpose or you can copy your entire application (typically the Adlib software folder including its subfolders), and use that *\data* folder if you want to have a test environment ready for the converted data. Instead, you can also use an existing *\data* folder as the target folder, if you already have a target environment set up (like a web application or Adlib test application). See steps 1-4 in the next chapter for details.

2. Adcopy also needs a target SQL database to copy your source database to. This can be a temporary, new SQL database (which you'll have to create) if you are doing a binary-to-XML conversion or if you want to create a separate database for testing purposes before restoring it elsewhere later. However, the target database can be a pre-existing SQL database as well: this will overwrite any existing data in that database. The advantage of this last option is that everything is already in place to start working with the new data immediately; the disadvantage is that if the target environment is live, you skip the important testing phase of the new database. See step 5 in the next chapter for details.

3. Execute Adcopy to actually copy the database and possibly apply some changes to the copy, like the (automatic) binary-to-XML conversion and/or the removal of some sensitive data from it. See step 6 in the next chapter for details.

4. If you have copied your database to another pre-existing database, you are done now, except maybe for some testing. If, on the other hand, you have copied your database to a new SQL database, you'll probably need to do some renaming of files and folders to turn the copy into a production/live database; before that, you could stil test the copy if you didn't just create a new data folder but copied your entire application to create a test environment.
   See steps 7-13 in the next chapter for details.

Note that an Adlib database definition is not the same as a SQL database definition. An Adlib database or index file are both tables in a SQL database, for example. Also, there is no separate priref table, but in SQL the priref does get indexed. See the *SQL Server and Oracle* document (**`Ctrl`**+click to go to the download page on our website), for more information about how Adlib databases reside in a SQL or Oracle database. If you first want to know more about how Adlib databases are structured in Adlib's proprietary CBF format, see the *Adlib software functionality profile* document, which can be downloaded as well.

3.2 Copy-converting your SQL database

1. For safety reasons, create a backup of your Adlib application and database. See the Installation guide for Museum, Library and Archive for information about making backups (and restoring them).

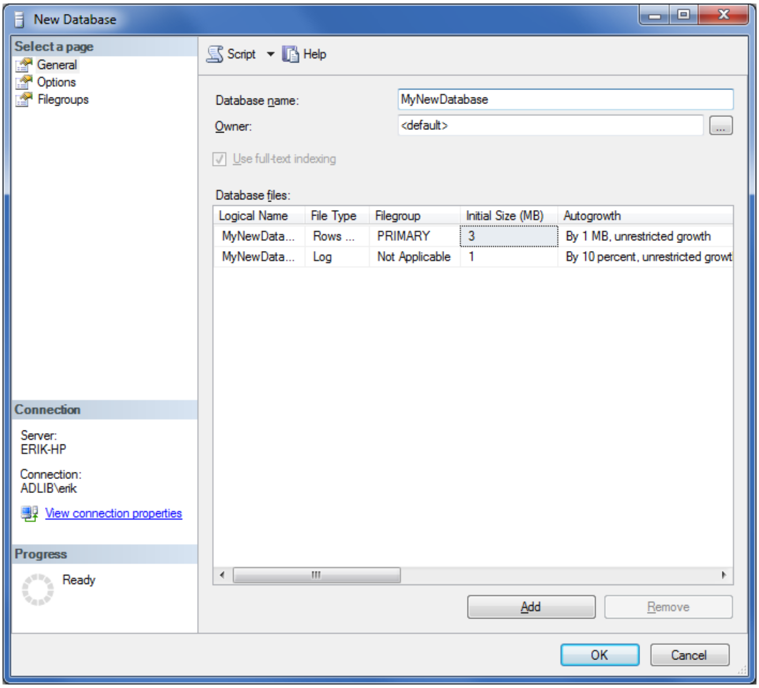
2. Copy the Adcopy files to a temporary folder on the local hard drive.

3. If you are about to convert binary data in your SQL database to XML data, or if you are about to copy your current SQL database to a new SQL database for other reasons, then in Windows Explorer look up the Adlib Software folder which has a *data* subfolder. This folder contains your database structure files (.inf). Underneath the Adlib Software folder, next to the *data* subfolder, create a new folder and name it e.g. *data.new*.
However, if you are about to insert a database copy into an existing database, then you don't need to create a new *data* subfolder: there will be an existing target *data* subfolder you can use.

4. If you created a new *data* folder in step 3, then copy all files from *data* into *data.new*. (The *.inf* files in the new folder will be changed later on.)
In all other cases you can skip this step.

5. From this step you can go two ways: (a) create a copy in a new SQL database, which is recommended for a binary-to-XML conversion, or (b) create a copy in another existing SQL database and overwrite any existing data in it.

    a. Open Microsoft SQL Server Management Studio (or a similar tool), and create a new database: right-click the *Database* node in the *Object Explorer* and in the pop-up menu choose *New database*... The name you enter for the *Database name* is not really important if you are doing the binary-to-XML conversion, because you'll change it later anyway. In the example below we named it *MyNewDatabase*. The credentials that Adcopy will use when you execute it next, need to have *datareader*, *datawriter* and *ddladmin* access rights to this database. Make any other desired settings before clicking *OK*.



Database name: MyNewDatabase

Owner: <default>

Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
MyNewData...	Rows ...	PRIMARY	3	By 1 MB, unrestricted growth
MyNewData...	Log	Not Applicable	1	By 10 percent, unrestricted growt

Server:
ERIK-HP

Connection:
ADLIB\erik

View connection properties

Ready

b. Open Microsoft SQL Server Management Studio (or a similar tool), and open the existing target database into which you want to create a copy. Make sure that the credentials that Adcopy will use when you execute it next, have *datareader*, *datawriter* and *ddladmin* access rights to this database.
Any existing data in this target database will be overwritten by Adcopy! If the target database is operational, this also means that your new copy will go live without having tested it. This always holds some risk, so if you take this path, make sure you have a backup of the target database ready to restore if the copy you create with Adcopy doesn't turn out the way you wanted it. Also be aware that while Adcopy is overwriting your live target database, it shouldn't be accessible to users at all, to prevent errors of all sorts: it's no use to bring the SQL database offline, because then Adcopy won't be able to write to it anymore either, so you would have to ask co-workers to stop working in the target database and/or present a temporary page on your website, for example.

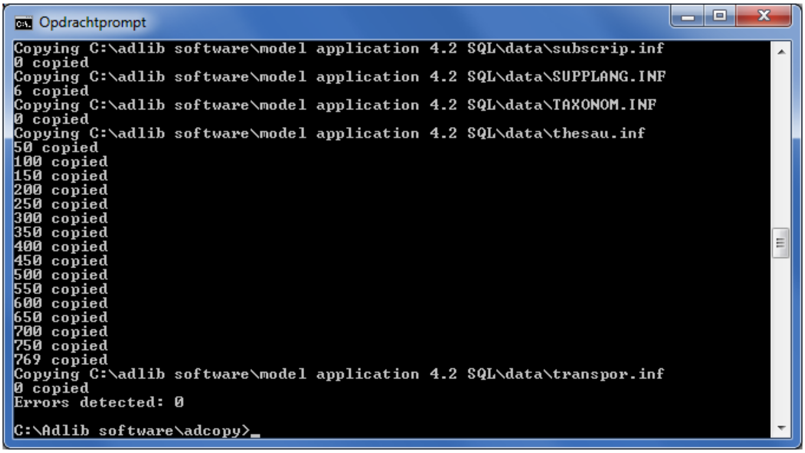
6. Adcopy can be controlled by command-line parameters, or by a parameter file in the same folder as the executable file, with the name *adcopy.xml*. If command-line parameters are provided then these will be used, otherwise the program will try to open the *adcopy.xml* parameter file.
   To provide command-line parameters, open a command line window and execute Adcopy using the following syntax:

   ```
   <(path to) adcopy> <path to the old data folder> <path to the
   new/target data folder> <the new/target database name> [the
   target SQL Server name] [user id for new/target database]
   [password for new/target database]
   ```

   Everything between [] is optional. If the new or existing target *\data* folder is not empty, any existing .inf files in there will be overwritten if copied files have the same name. If you are copying your database to another existing database, you can use the existing *\data* folder of the latter as the target data folder. The destination SQL Server name is optional: by default it is the (local) server, so only if your new database is destined for a different SQL Server, you will have to provide it explicitly. The name of the destination server will be stored in the copied .inf files. The user id is the user name with which Adcopy connects to the SQL server; supplying the user id in this way is only possible if the SQL server uses the SQL server authentication mechanism. This user name will also be stored in the copied .inf files. If a user id is provided then a password is also required, which will be stored in the copied .inf files as well. Further, the user id and password will be copied from the original .inf files if no new user id and password are provided. Example:

   ```
   adcopy “C:\adlib software\model application 4.2 SQL\data”
   “C:\adlib software\model application 4.2 SQL\data.new”
   MyNewDatabase SQLserver2008\Adlib
   ```

   This will take a while to complete. When it is finished, any errors will be reported in the command prompt window.



```
Copying C:\adlib software\model application 4.2 SQL\data\subscrip.inf
0 copied
Copying C:\adlib software\model application 4.2 SQL\data\SUPPLANG.INF
6 copied
Copying C:\adlib software\model application 4.2 SQL\data\TAXONOM.INF
0 copied
Copying C:\adlib software\model application 4.2 SQL\data\thesau.inf
50 copied
100 copied
150 copied
200 copied
250 copied
300 copied
350 copied
400 copied
450 copied
500 copied
550 copied
600 copied
650 copied
700 copied
750 copied
769 copied
Copying C:\adlib software\model application 4.2 SQL\data\transpor.inf
0 copied
Errors detected: 0

C:\Adlib software\adcopy>_
```

As mentioned, when no command line parameters are supplied, Adcopy will try to find and read *adcopy.xml.* This file provides the same parameters as the command line, but it offers a few additional options. Here is an example of the content of the *adcopy.xml* parameter file:

```
<?xml version="1.0" encoding="utf-8"?>
<AdCopyJob xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SourceFolder>C:\adlib4.2-SQLserver\data</SourceFolder>
  <!-- folder to copy data from -->
  <DestinationFolder>C:\adlib4.2-SQLserver\data2</DestinationFolder>
  <!-- folder to copy data to -->
  <DestinationDatabase>model42t</DestinationDatabase>
  <!-- name of the destination database -->
  <DestinationServer>(local)</DestinationServer>
  <!-- name of the destination server -->
  <UserId></UserId>
  <!-- userid to be used and placed in the destination .inf file -->
  <Password></Password>
  <!-- password to be used and placed in the destination inf file-->
  <MileStone>50</MileStone>
  <!-- milestone value -->

  <!-- a list of databases with fields that need to be removed -->
  <DatabaseList>
    <Database Name="collect">
      <!--entry for each database of which fields must be removed-->
      <Exclude>
        <!-- list of Adlib tags that need to be removed -->
        <string>ls</string>
        <!-- one string entry per tag that needs to be removed -->
        <string>VY</string>
      </Exclude>
    </Database>
  </DatabaseList>
</AdCopyJob>
```

The additional options in the parameter file are:

- the milestone value, to indicate progress during the conversion;
- an optional list of one or more Adlib databases, each containing a list of tags which need to be stripped from the copy. Of linked fields to strip, you only need to provide the link reference tag. This functionality operates on the unresolved data as it is stored in the record, which means without data in the linked field itself or in any merged-in fields. So, removing the link reference tag from the database copy will leave out the link reference and the relevant linked data including merged-in data. To strip tags from more than one database you must repeat the `<Database>` element with another name and accompanying list of tags.
  This functionality is handy when you want to create a copy of your database without any sensitive information in it, to be applied as the database for your public web application for instance, or for use by third parties. Typically you would not use this option if you just want to convert binary data in your SQL database to XML data.

7. If you have copied your database to another existing database, you are done now. Just check the target environment to see if everyting is as it should be. Skip steps 8 and further.
   If you haven't been copying your database to another pre-existing database and if you created a new subfolder for the .inf files in step 3, then rename the old \data folder in Windows Explorer, for instance to \data.old, and subsequently rename the new subfolder (e.g. \data.new) to \data. The changed .inf files now reside in your current \data folder.
8. In Microsoft SQL Server Management Studio, rename the old database (let's say this was called *AdlibDB*), for instance to *MyOldDatabase*. Also rename the new database (in our example *MyNewDatabase*) to the original name of the *MyOldDatabase* database: *AdlibDB* in this example. So the old *AdlibDB* has now been exchanged for the new *AdlibDB* which was created by Adcopy.
   (To rename a database, right-click it and choose *Rename* in the pop-up menu. You can only rename a database if it is not in use.)

9. If the old database, *AdlibDB* in our example, had specific access rights settings, then apply the same settings to the new database now.

10. Start Adlib Designer, open your Adlib application in the *Application browser* and select one of the Adlib database structure files (.inf). On the *Database properties* tab, change the adjusted *Data Source Name* (*MyNewDatabase* in our example) to the original name (like *AdlibDB* in this example) and leave the field. Adlib Designer will ask you if you want to apply the change to all databases: click *OK*. Then click the *Test* button to the right of this option, to test the connection. Only if the test succeeded, click the *Save all modified files* button and save all changed .inf files.

11. Start your Adlib application and check if all is well.

12. If you tested your new database succesfully, you may delete the *\data.old* folder from your system; make sure you do not remove the *\data* folder!

13. In Microsoft SQL Server Management Studio you can also delete the old database by right-clicking it (*MyOldDatabase* in our example) and choosing *Delete* in the pop-up menu. You can only delete a database if it is not in use.)

By the way: Adcopy is backwards compatible, so an older adlwin.exe can be used to open the database again after being processed by Adcopy.

4 Adsearch.exe

Adsearch.exe is an Adlib command-line program with which you may execute search queries in an Adlib CBF database (no support for SQL: use an API instead), from within a command line window. The result will be written to a pointer file which you can use normally in Adlib. The syntax is as follows:

```
[directory\]adsearch -l <[directory+]database_name> -p <point-
er_file_number> [-t "<pointer_file_title>"] <[directory\]file_name>
```

The possible options are:

-h	`adsearch -h` displays Help similar to this document.
-l <[directory+]data-base_name>	(This is the lowercase letter L.) Behind a space, fill in the name of the Adlib data-base you want to search, preceded (if necessary) by the path to it and a plus sign. This option is mandatory.
-p <pointer_file_number>	Behind `-p` and a space, enter the number of the pointer file to which the result must be written. If you enter `-p 0`, then adsearch assigns a pointer file number for you which is not yet in use. This option is mandatory.
-t "<pointer_file_title>"	Behind `-t`, provide the name of the tar-get pointer file, enclosed by double quotes.
<file_name>	You must specify the search query in a text file and save it with the *.src* exten-sion. The syntax of the search query must be equal to that of the Adlib *Expert search system*. Use tags or English field names. From version 6.4 you may save this file in UTF-8 encoding, but you should use ANSI encoding in older ver-sions. At the end of the adsearch command you must enter the name of this file, without extension. Enclose the name by double quotes if there are any spaces in it. This option is mandatory.

If, in the command line window, you are in the directory in which *adsearch.exe* is located, for instance the \executables directory, then you can enter the following command, for example:

```
adsearch -l ..\data+Collect -p 1 -t "Object is postcard" objectnam-
equery
```

You must have already created the *objectnamequery.src* file (in the current folder) though. This may contain the following search query, for example:

```
object_name = postcard
```