ResourceMiner
Manual

Understand,
Develop & Document

version 3.2
2014-07-18

Table of Content

General

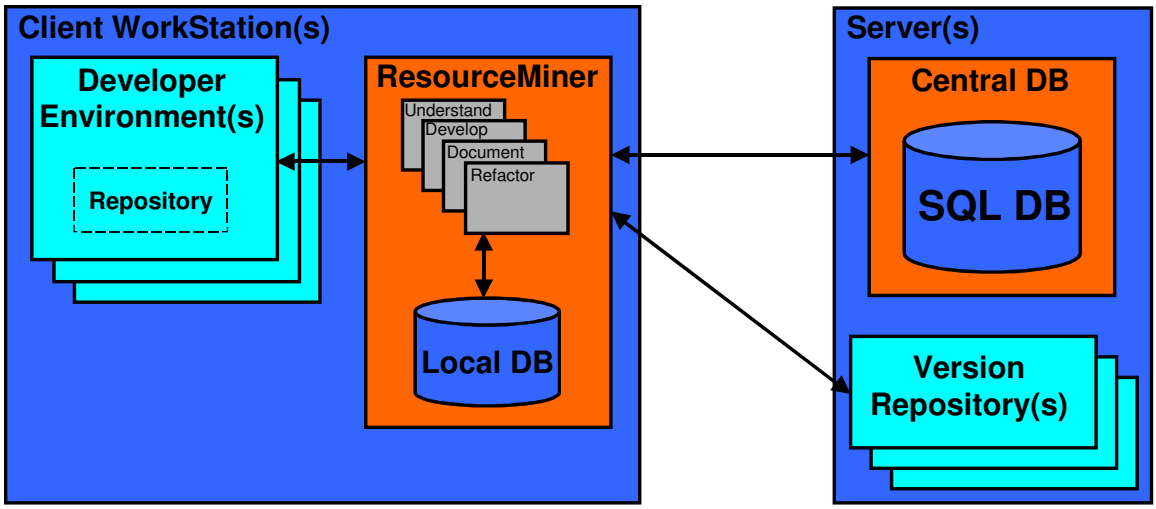
ResourceMiner is a tool for source code analysis to enhance the productivity of software development and maintenance. The tool supports a process where source code analysis and change decisions are separated from changes in the source code.

Source code is loaded (parsed) into ResourceMiner databases using different language definition extensions. ResourceMiner has support for several software languages/environments and new language definition extensions are being developed continuously and on demand.

Product package

1. **Understand**
   Is the base module which contains the database, load functionality, View- and Query forms. A system developer spends up to 50% of the time to analyze and understand existing source code to be changed or extended, *Understand* reduces this time.
   Understand is the base which all other modules need to be able to function.
2. **Develop**
   This extension contains functions to create a WorkList and BizAnalysis, support for integration with different development environments (ie Mainframe, Unix, Repositories and Editors) and give possibilities to define static or dynamic connections (middleware) within the source code.
3. **Document**
   This extension contains integration with Visio to generate diagrams (boxes, lines and text) which can be customized to different types of drawing standards. Also includes Measure functionality and extensions to automate analysis by macro coding in Excel, named OnTopApps.
4. **Refactor**
   This extension contains a number of functions that enables automatic code changes like mass-changes, restructuring, language change and more.
   Refactor is not described in this manual.

ResourceMiner architecture and integration with developer environments and repository's:



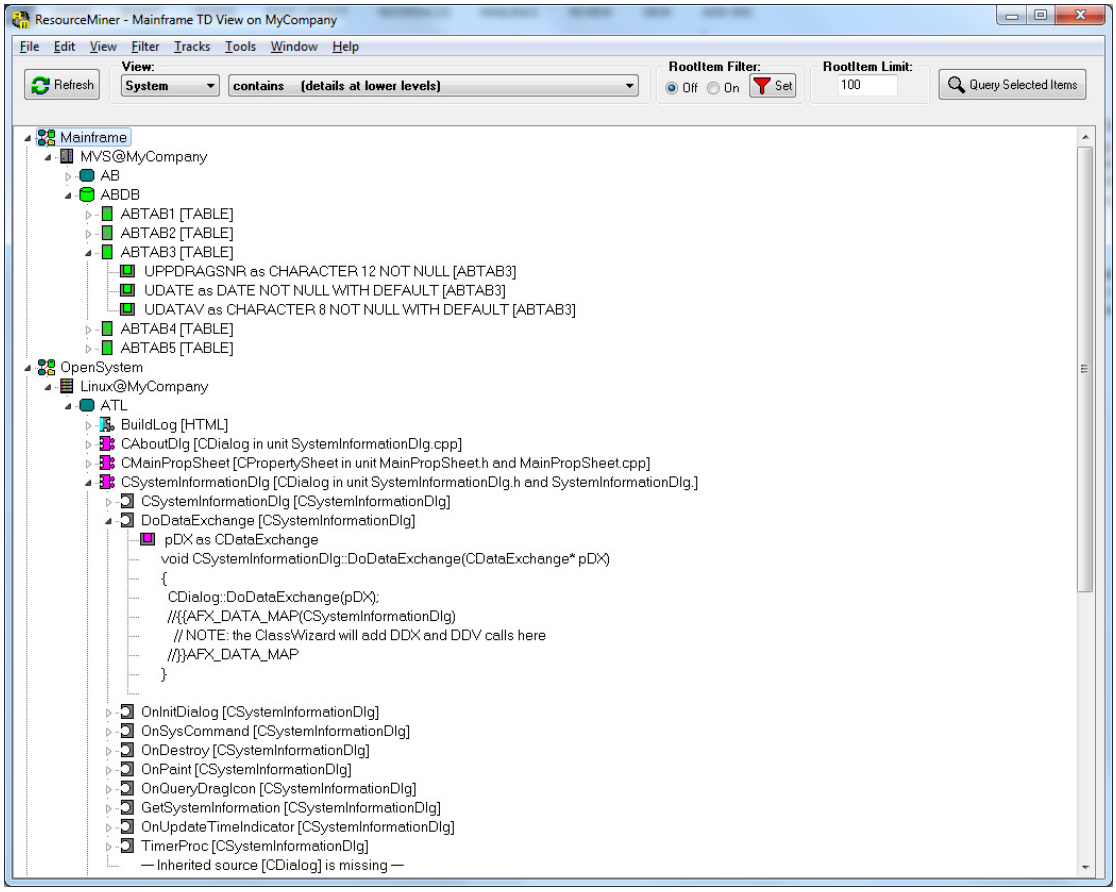
Version history

Read the *What_is_new_in_RMnnnn.doc* in COMMON_DOCUMENTS\ResourceMiner\Logs to trace changes and enhancements for this software.

Understand

The main forms in Understand are View and Query which can be considered as living documents of a system. View and Query can be saved as rmv- and rmq-files, it is possible to double click these files to make ResourceMiner start and show the content. ResourceMiner can also be started as a normal application on the start menu or by its icon on the desktop. As default one View-form and one Query-form opens.

View



View gives the opportunity to see one or several collaborating systems from different views. The view contains *Items* and its relationship to other Items (the system OpenSystem is an Item, so is also the object CAboutDlg). Each View has one or more *rootitems*. Rootitems are the leftmost items that the view starts with. Items can be expanded by clicking its plus/triangle-sign to the left. Expansion will show one or more related items which also can be expanded. When the plus/triangle-sign is missing, the item can not be expanded. Lines between Items are their *relation*.

Control panel

At the top of the View-form there is a panel to control what type of items to show, what kind of relations to show, a filter button to choose which rootitems to work with and a query button to find source code statements within the items that the View is showing. The list of rootitems is limited to get fast response time, the limit can be changed.

Refresh
Changes in the control panel are not reflected in the View-form until the *Refresh* button is clicked.

View
Choose which level of detail (level of abstraction) to follow. The choices are System, Computer, Application, Package, Object, Entry, Code or Data.

Relation and Direction
Distribution is a relation showing how the source code is distributed among system, computer, application, package, object and entry. 'contains' show *Distribution TopDown* and 'is part of' show *Distribution BottomUp*.

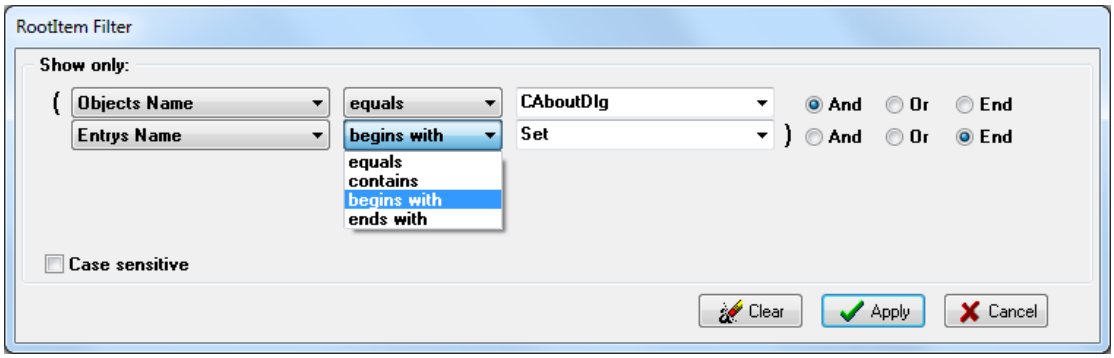
Dependencies is the other relation showing how system, computer, application, package, object and entry is dependent of each other. 'uses' show *Dependencies TopDown* and 'is used from' show *Dependencies BottomUp*.

Show in Execution Order (applies only to Dependencies)
The combination Object Uses and Entry Uses gives the opportunity to *Show in Execution Order*. This means that the expanded Items will show up in the order they are used. Same Object/Entry can show up several times. For the combination Entry Uses *Show in Execution Order* is the default because it will show a callchain in order of execution which is a common use case.

When *Show in Execution Order* is not selected or not visible, the expanded items are shown in name order and not in order of execution. Each dependent item is showed only once; showing that the item is used at least once.

RootItem Filter
There is a filter to limit the list of rootitems. Rootitems are the items the user is interested to know more about. The filter applies only to rootitems, not what is expanded from the rootitems.

The button *Set* (with a red funnel) will open this window:



Filter can be set on all levels of details, for example system, object and entry. For each condition of *And* or *Or*, the form expands with another row for additional conditions, which gives possibilities to advanced filters. The choice *End* means that the filter condition ends, it is possible to shorten a filter condition by clicking *End* at the row where it should end.

The chosen level of detail affect what the filter can exclude. If View is *Object* and the filter contains a condition for Entrys (see picture above), the Entry-condition will not apply. The condition for entry will apply when changing View to *Entry*, *Code* or *Data*.

Clear removes the filter. *Apply* set the filter active and closes the window. If the filter is cleared when clicking apply, the filter is inactive.

RootItem Limit
To increase performance for View-form updates, there is a limitation of the number of rootitems to show. The default is maximum 100 rootitems but this can be changed by the user.

When the rootitems are limited, this text is showed at the top of the View-form:
*** RootItem Limit reached, only the first 100 hits in the database showed ***

Observe that when the rootitems are limited they are controlled by the physical order in the database and not by name or filter conditions. To be sure to see all items beginning with for example 'A', use filter to avoid limitation (or increase rootitem limit to be shore all are shown).

Query Selected Items
When at least one Item is selected, this button will open a new Query-form and set the search filter to the selected item(s) or Track (see below).

Menu for View- and Filter settings
The main menu contains the View-form specific menus *View* and *Filter*, which gives an alternate way of controlling the view- and filter settings. The choices are the same as in the control panel.

Tracks
Tracks are a selection of related Items that the user is working with. A track can exist in all view types and represent for example a 'contains'-track or 'is used from'-track. Form a track by selecting at least two directly related items by left click the first item and then use Shift+leftclick or Ctrl+leftclick to select the other related item(s).

Tracks are used to query a delimited amount of source code in the Query-form, normally to find statements in a callchain or in a number of objects related to each other. Track is also used to generate diagrams in Visio (requires the extension Document).

Tracks can be added as bookmarks to be able to jump between different Tracks easily.

Tracks functions exist in the main menu under *Tracks*, the functions in the menu are:

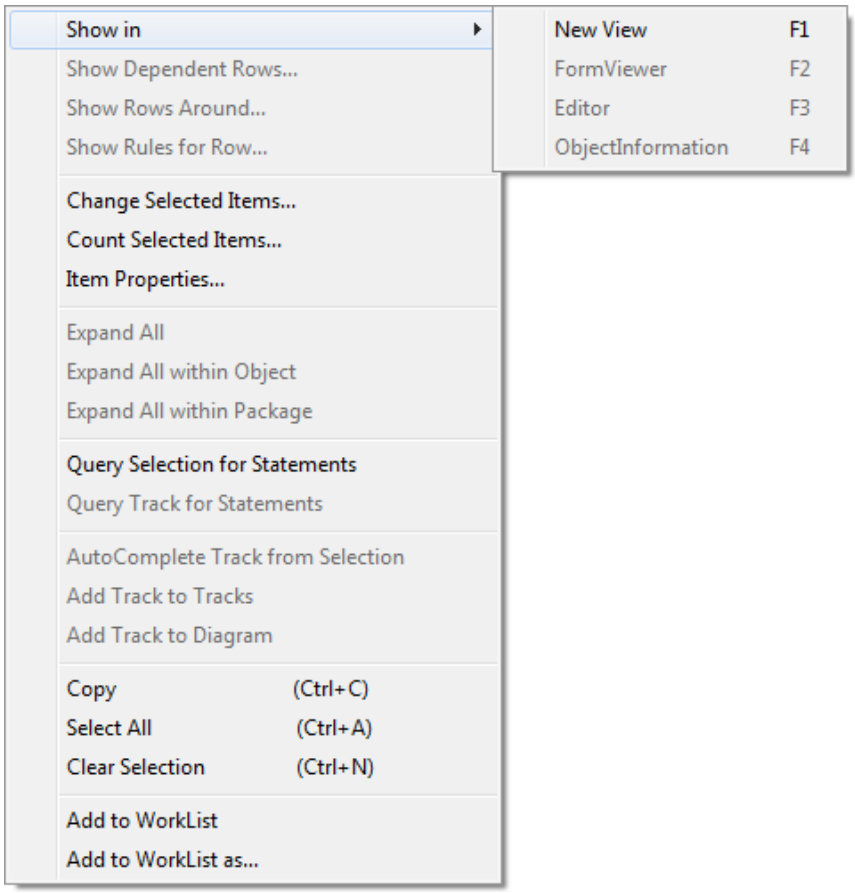
Menu choice AutoComplete Track
A valid Track contains at least two directly related items. If a number of items is selected but not directly related it is possible to choose *AutoComplete Selected Track*. This menu choice is enabled only when at least two items are selected and the items do not form a valid Track.

Menu choice Add Track
Saves a Track to the Tracklist. This menu choice is enabled only when a valid Track is selected.

Tracklist
The menu *Tracks* contains the tracklist and show saved tracks, the name of a saved track is automatically generated from its first and last item names. By choosing a previously saved track from the menu, all items in the chosen track are selected.

Popupmenu for View

The View-form has a popupmenu with a number of choices which is enabled depending on what item selection(s) is made in the View-form:



Show in New View
At least one Item must be selected. Opens a new View-form and sets its rootitem filter to the selected item(s). If more than one Item is selected, they must be of the same level of detail.

Show in FormViewer
Only one Item must be selected and this Item must be a visual object (blue Icon). A new window shows how this visual object looks to the user with name on variable fields that can change content during program execution. This function is implemented for 3270 like screens and separate language definition extensions are needed.

Show in Editor
Only one Item must be selected. This choice is part of the extension *Develop*. If Develop is installed, the source file is opened with the application which is linked to the filetype in Windows or the application given in *Tools/Settings/Editors*.

Show in ObjectInformation
Only one Item must be selected, and it must be an object. ObjectInformation is a separate window showing all kinds of information about an object (see separate description below).

Show Dependent Rows
Two directly related Items expanded in View Type Dependencies must be selected. A new window shows the lines of code that makes the items relate. Use to see dependencies at code level for detail levels above View By *Code*.

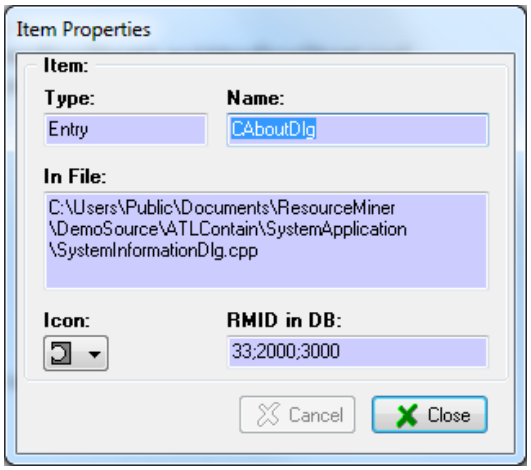
Show Rows Around
Only one Item that contains code must be selected. Show the lines of code before and after in a new window. The window is scrollable with the PageUp/PageDown-buttons within its source file.

Show Rules for Row
Only one Item that is a line of code must be selected. A new window shows the conditions in order of execution that must be passed to make the row execute.

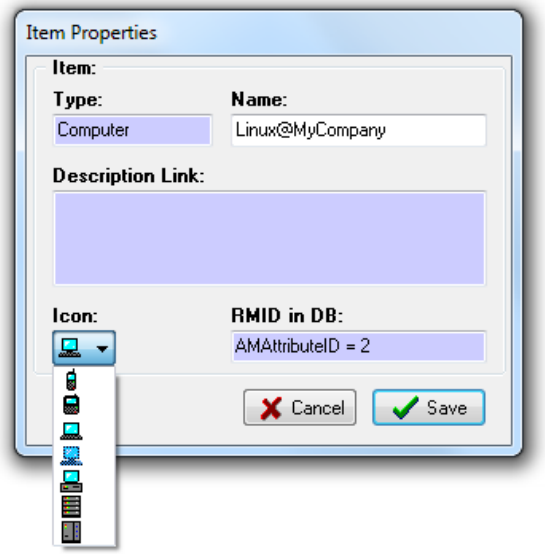
Change Selected Items
Gives the opportunity to select or deselect all visible items with similar names.

Count Selected Items
At least one Item must be selected. Show how many items that are currently selected.

Item Properties
Only one Item must be selected. Show the properties for the selected Item. For Package, Object, Entry, Code and Data its source file path and name is shown, the information can not be changed.



For System, Computer and Application *Name* and *Icon* can be changed:



System, Computer and Application name and icon do not reside in any source file, the names are given in the Load-form (see below) and the default icons given at load can be changed here.

Expand All
Only one Item that can be expanded must be selected. Expand all related items recursive.

Expand All within Object
Only one Item that can be expanded must be selected. Expand all related items recursive within same Object.

Expand All within Package
Only one Item that can be expanded must be selected. Expand all related items recursive within same Package.

Query Selection for Statement
At least one Item must be selected and the selected items must not form a Track. Opens a new Queryform and sets the search filter to the selected item(s). The button 'Query Selected Items' will give the same function when Items not forming a Track is selected.

Query Track for Statement
A Track must be selected. Opens a new Queryform and sets the search filter to the selected Track. The button 'Query Selected Items' will give the same function when Items forming a Track is selected.

AutoComplete Track from Selection
A valid Track contains at least two directly related items. If a number of items are selected but not directly related it is possible to choose *AutoComplete Track from Selection*. This menuchoice is enabled only when at least two items are selected and the items do not form a valid Track.

This is same function as the mainmenu choice *Tracks/AutoComplete Track*.

Add Track to Tracks
Saves a Track to the Tracklist. This menuchoice is enabled only when a valid Track is selected.

This is same function as the mainmenu choice *Tracks/Add Track*.

Add Track to Diagram
A Track must be selected. This choice is part of the extension *Document*. If Document is installed, the Track is added to current diagram in Visio.

Copy
This is same function as the mainmenu choice *Edit/Copy*.

Select All
This is same function as the mainmenu choice *Edit/Select All*.

Clear Selection
This is same function as the mainmenu choice *Edit/Clear Selection*.

Add to WorkList
At least one Item must be selected. This choice is part of the extension *Develop*. If Develop is installed, the selected item(s) is added to current WorkList.

Add to WorkList as
At least one Item must be selected. This choice is part of the extension *Develop*. If Develop is installed, the selected item(s) is added to current WorkList with the extra information given by user in a dialog that shows up.

Items in View

There is a number of itemtypes in the View-form. Languages has different subset of those types.

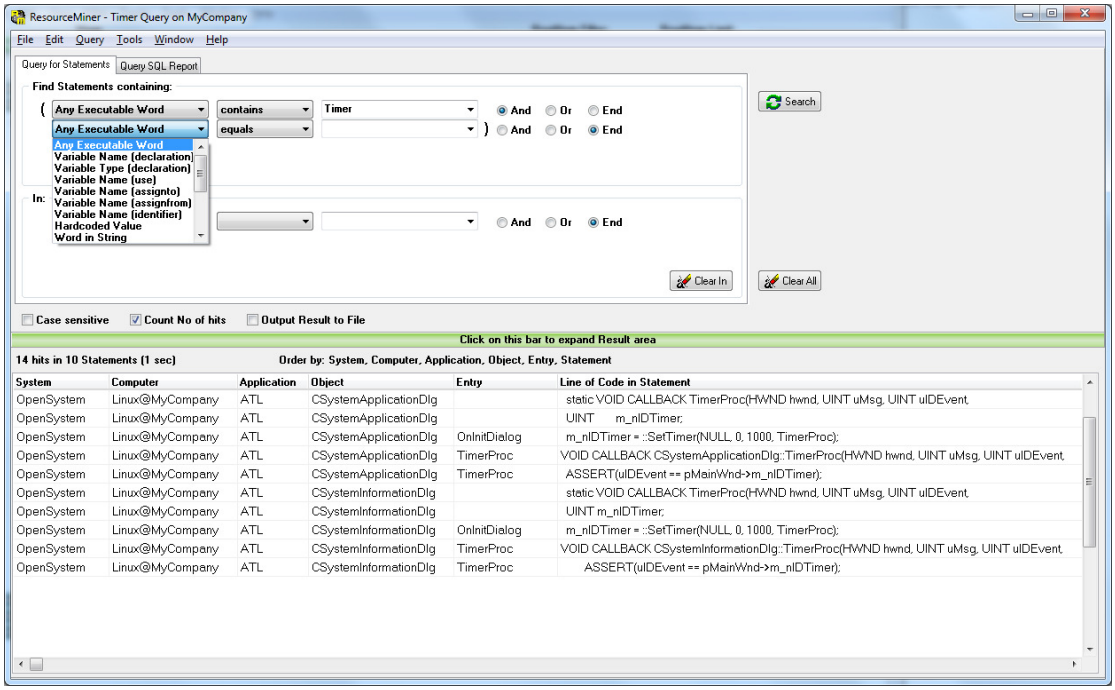
Item	Description
System	
<name>	Name of a system, the name is given in the Load-form
Computer	
<type of computer>	The type is given in the Load-form, for example Client@ABC, the icon can be changed to reflect the type of computer.
Common Source for …	Common source for several computers within same system, is given in the Load-form by omitting both Computer and Application.
Application (subsystem or tier)	
<name>	Name of application/subsystem/tier/database, the name is given in the Load-form. The icon can be changed to reflect application or database.
Common Source for …	Common source for several applications/subsystems within same computer, given in the Load-form by omitting Application.
Package	
<name> [<type>]	A package contains one or more objects. In CSP the package equals a program, in Delphi it is a BPL, in VB/.NET it is a DLL etc.
Object	
<name> [<type>]	Programobject that are visible on screen, for example a 3270-map or a Form
<name> [<type>]	Subobject that are visible on screen, but needs a form to be in. This type of object is also called a visual control or visual component.
<name> [<type>]	Programobject that are invisible to the user.
<name> [<type>]	Programobject that is an internal table.
<name> [<type>]	Programobject that is a record or struct. In some languages these must be declared within another programobject, in other languages they can be declared separate.
<name> [<type>]	Programobject that must be included in other programobjects to be able to execute. Is also called copybook or includefile depending on the language.
<name> [<type>]	Programobject that is a Table or a View in a database.
<name> [<type>]	Programobject that is a record static connected to persistent data in a database table or file.
<name> [<type>]	Programobject that is a file.
Entry	
<name> [Object]	Entry to executable code that can be called. Entry is the same as a method, function, process, procedure, section etc depending on language.
Code	
<code> [Object.Entry]	A line of code containing one or more calls has this icon. If it has a plus/triangle-sign to the left it can be expanded to see the code in the called Entry(s).
Data	
<name as type> [Object.Entry]	Variable, ie a holder for data
<name as type> [Object.Entry]	Field, ie a holder for data that is static connected to persistent data in a database table or file.

Query

Query makes it possible to search ResourceMiner databases in two different ways; *Query for Statements* and *Query SQL Report*.

Menu for Query
The mainmenu contains the query-form specific menu *Query*. The choices are *Search* and *Clear Query* which is the same as the buttons *Search* and *Clear All*.

Query for Statements



The tab *Query for Statements* is used to search for words in statements, the result is matching lines of code and information about where they reside. For each condition with *And* or *Or*, the *Find*-section expands with another row for additional conditions, which gives possibilities to advanced search. The choice *End* means that the search condition ends, it is possible to shorten a search condition by clicking *End* at the row where it should end.

The scope of the search can be narrowed by the *In*-section to a certain system, computer, application, object etc. For each condition with *And* or *Or*, the *In*-section expands with another row for additional conditions. The *In*-section can be cleared with the button *Clear In*.

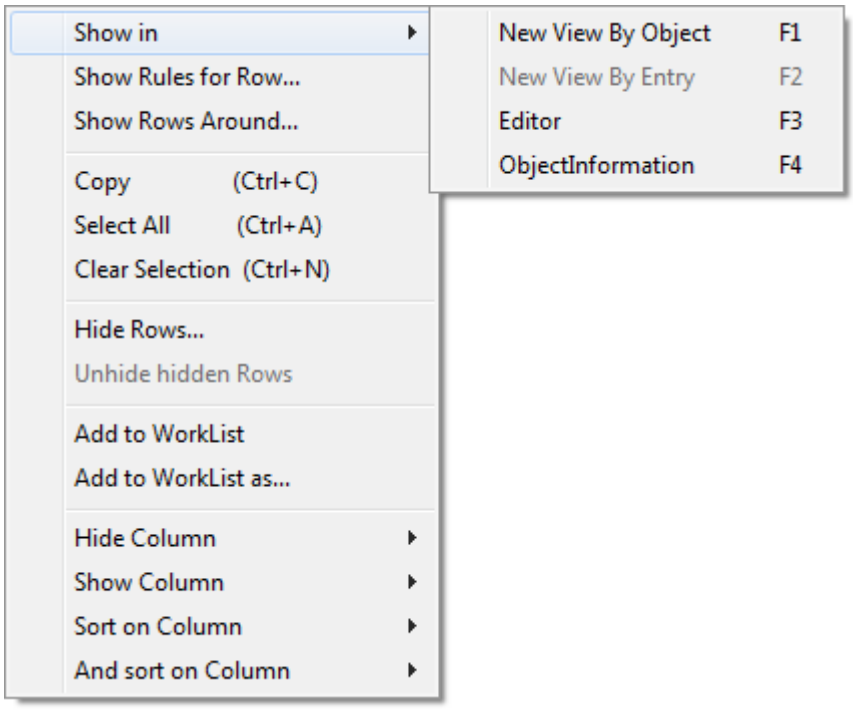
The button *Search* executes the query created by the conditions in *Find* and *In*, the whole query can be cleared with the button *Clear All*.

The result of a query can be many rows. To increase performance, only the first 40 rows will be shown in the resultlist. Each time the user scrolls down to the end of the list, another 40 rows will show up and so on until the resultlist contains all rows. By selecting *Count No of Hits* it is possible to know about how many rows the resultlist finally will contain when scrolled to the end.

The resultlist has several columns and the sort order can be changed. Unwanted columns can be hidden and rows of interest can be further analysed via functions in a popupmenu (se below). The resultlist can be expanded to cover the whole Queryform by clicking the green bar.

Popupmeny for Query for Statements

The resultlist in *Query for Statements* has a popupmenu with a number of choices which is enabled depending on what row selections is made in the resultlist:



Show in New View by Object
Opens a new View-form with the filter set to show the Object(s) where the selected row resides.

Show in New View by Entry
Opens a new View-form with the filter set to show the Entry(s) where the selected row resides.

Show in Editor
One row only must be selected. This choice is part of the extension *Develop*. If Develop is installed, the source file is opened with the application which is linked to the filetype in Windows or the application given in *Tools/Settings/Editor*.

Show in ObjectInformation
One row only must be selected. ObjectInformation is a separate window showing all kinds of information about an object (see separate description below).

Show Rules for Row
One row only must be selected. A new window shows the conditions in order of execution that must be passed to make the row execute.

Show Rows Around
One row only must be selected. Show the lines of code before and after in a new window. The window is scrollable with the PageUp/PageDown-buttons within its source file.

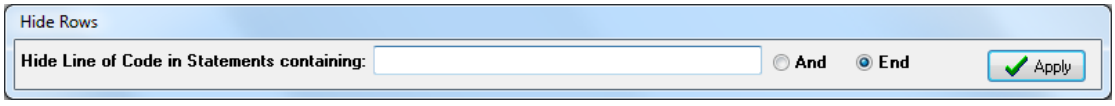
Double-clicking on a row will start this function. It is a frequently used function.

Copy
This is same function as the mainmenu choice *Edit/Copy*.

Select All
This is same function as the mainmenu choice *Edit/Select All*.

Clear Selection
This is same function as the mainmenu choice *Edit/Clear Selection*.

Hide Rows
Makes it possible to take away unwanted rows from the resultlist.



Unhide hidden Rows
Hidden rows can be brought back with this function.

Add to WorkList
At least one row must be selected. This choice is part of the extension *Develop*. If Develop is installed, the selected item(s) is added to current WorkList.

Add to WorkList as
At least one row must be selected. This choice is part of the extension *Develop*. If Develop is installed, the selected item(s) is added to current WorkList with the extra information given by user in a dialog that shows up.

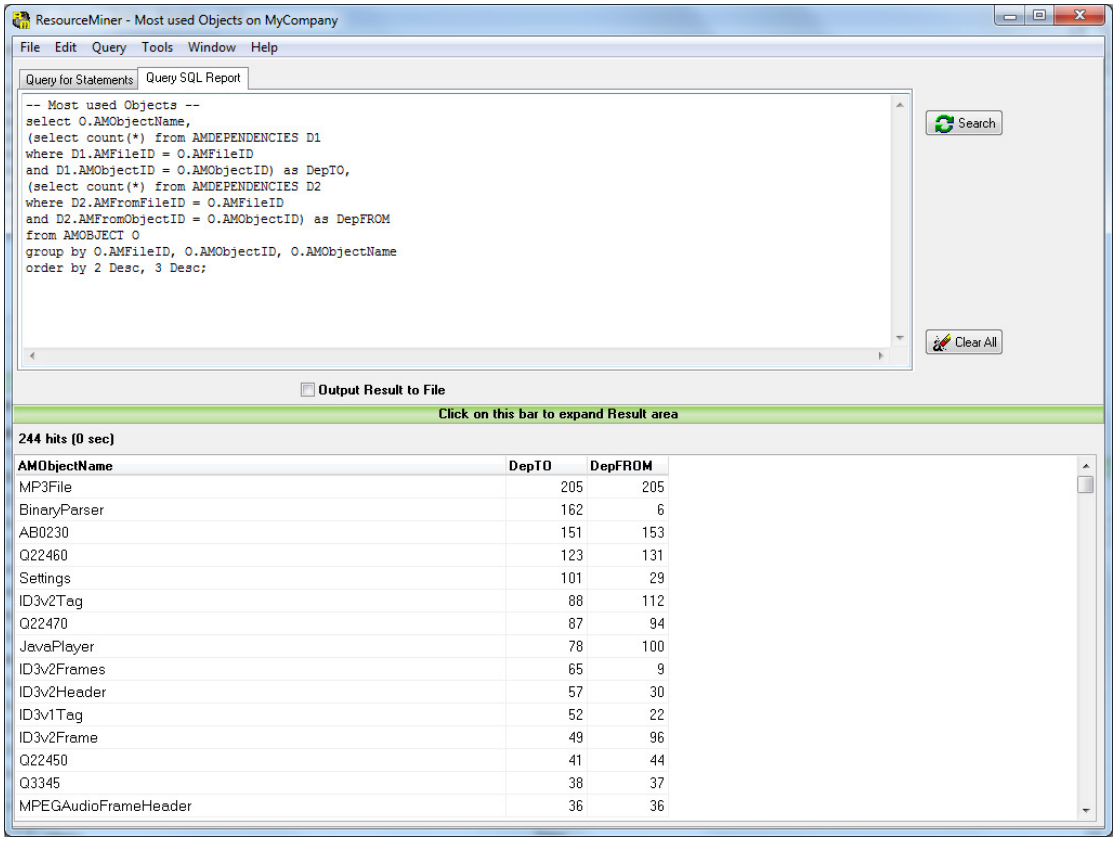
Hide Column
A list of visible columns, hides the column of choice.

Show Column
A list of hidden columns, shows the column of choice.

Sort on Column
A list of sortable columns, sorts the result on the column of choice (replaces current sort order).

And sort on Column
A list of secondary sortable columns, sorts the result on chosen column too
(expands current sort order, disabled if current sort order is missing).

Query SQL Report



The tab *Query SQL Report* is used to write and execute free SQL-queries to ResourceMiner databases. A number of predefined reports (see below) can be used, altered and saved.

The complete result is always shown, not 40 rows at a time as in *Query for Statements*.

Query SQL Report do not have any popupmenu. The resultlist can be expanded to cover the whole Query-form by clicking the green bar.

Predefined Query SQL Reports

The mainmenu choice *File/Open Query SQL Report* will show predefined reports to choose from.

Those reports show a lot of possibilities and can give inspiration to much more. The reports can be adjusted, saved and renamed to own needs. Here is a description of the predefined reports:

- **All declared Variables.rmq**
  Overview of all used variables.

  Useful for:
  Checking naming conventions for variables

- **All used Components.rmq**
  Gives an overview of all used classes in the code.
  In the query used here there is a part "and AMWord like 'J%'"
  This is used to identify only the standard classes in eg Java. When the targeted language uses another name convention or doesn't use one at all either adjust the sql statement or remove this part to get all used classes.

  Useful for:
  Identify any 3rd party class libraries used

- **C# Coding Standard for ASPNET Components.rmq**
  Gives an overview of all standard ASP.NET components that are used.
  E.g. in an ASP.NET webproject it will give all textboxes, dropdownlists etc that are used.

  Useful for:
  See if any advanced standard components are being used or only simple components.

- **C# Coding Standard for Class and Method Names.rmq**
  Overview of methods for each class.
  Also information about IN/OUT parameter types for these methods.

  Useful for:
  Checking name conventions for methods and parameters, see if there are possible complex methods when there are any large signatures (methods with many parameters)

- **C# Coding Standard for Class Variables.rmq**
  Overview of all variables and properties (datatype and name) for each class.

  Useful for:
  Checking for complex/extended classes (with a lot of variables or properties), also checking naming conventions used.

- **C# Coding Standard for Method Variables.rmq**
  Overview of all variables and properties (datatype and name) for each class that are being used inside methods of these classes.

  Useful for:
  What kind of variables are being used (complex or simple, native or user defined). Also checking naming conventions used.

- **CallChain StartPoints.rmq**
  Overview of possible entry points in the code for each class, points where the codeflow can start.

  Useful for:
  Seeing possible starting points in the code, many starting points or just a few ?

- **CheckDuplicateObjectNames excluding Subobjects.rmq**
  Overview of duplicate object names that may refer to multiple versions of object definitions. These can give a more complex idea of the code than it actually is.
  Subobjects for .NET and other GUI languages are components on a Form.
  They often have duplicate names, e.g. TextBox1 on several forms.

  Useful for:
  See if any unused objects can be removed to get a clearer view on the complexity of the code.

- **CheckDuplicateObjectNames.rmq**
  Overview of duplicate object names that may refer to multiple versions of object definitions. These can give a more complex idea of the code than it actually is.
  For .NET and other GUI languages, use the script above (excluding subobjects) to get a clear answer. This report (including subobjects) is good for procedural languages without GUI, eg Mainframe Cobol.

  Useful for:
  See if any unused objects can be removed to get a clearer view on the complexity of the code.

- **CheckParseErrors.rmq**
  Overview of any parse errors in the code.

  Useful for:
  Checking if the code on itself is ready to be compiled, or needs adjustments.
  Errors can also be caused by misinterpretation of the code file extension (see settings for that).

- **Count FP with Backfiring.rmq**
  FP stands for Function Point. It is normally a complex algorithm to count Function Points for an Application. The backfiring method consists of a set of factors for converting between Function Points and LOC. The factors can be changed by the user if needed.

  Useful for:
  Estimating the size of an application. By measure between versions of an application, the growth of the application is measured. By dividing the FP growth with the hours spent on developing, the productivity can be measured (FP/hour). The backfiring method has drawbacks but can be useful anyway. ResourceMiner has an add-on module that can count FP without the drawbacks of backfiring.

- **Count COMMENTS.rmq**
  Gives the number of comments (lines, not blocks) that are found in the code.

  Useful for:
  Checking if enough documentation is given throughout the code (compared to the number of LOC's)

- **Count ESLOC.rmq**
  Gives the number of ESLOC's in the code.
  ESLOC = Effective Source Lines Of Code
  It is simply the number of lines with code minus the number of comment lines.

  Useful for:
  Just another way of measure the size of an application. By dividing number of comment lines with ESLOC or SLOC you will have the density of comments. It is a way to quality control that system developers write enough comments.

- **Count SLOC.rmq**
  Gives the number of SLOC's in the code.
  SLOC = Source Lines Of Code

  Useful for:
  Many lines means a complex project with a lot of functionality, or cumbersome code.

- **Distinct Missing Entrys.rmq**
  Overview of not defined entries/properties.
  Note: When trying this on an ASP.NET (C#) project you may get a lot of results, like 10px in a CSS style definition "font-size:10px", or "family" in "font-family: Verdana", or "_blank" in a target definition for a "<a href…" definition.
  This is normal because of the behavior of the C# parser. You can ignore hits that obviously are not an Entry. Be aware that properties are Entrys. The result of this query is normally a long list. It shows all used Entrys and Properties of the class library and 3:rd part libraries. You should not find any of the Entrys/Properties of your own source here, but that is cumbersome to check though. It's better to use the 'Distinct Missing Objects.rmq' only.

  Useful for:
  Checking if all entries are defined

- **Distinct Missing Objects.rmq**
  Overview of not defined objects in code.

  Useful for:
  Identifying missing source code

- **McCabe Complexity Count.rmq**
  Overview of complexity values for each entry point
  Like Function Points, McCabe Complexity is a standard measurement. It measures the number of execution ways in the source, i.e. it counts all IF, CASE, WHILE etc.
  This report count per Entry. If an Entry has no conditions at all (no IF, CASE, WHILE etc) it has only one way of execution and hence the complexity 1. With one IF-statement it would be 2. Good programming standards says that Entrys with complexity above 15 (or so) is not good, but it is OK with a high number because of a lot of short well structured CASE or ELSE IF statements.

  Useful for:
  Identifying the most complex parts of the code.
  Can be used to monitor development to avoid complex programming.

- **McCabe Total.rmq**
  Sum of all complexity values in the "McCabe Complexity Count" script
  Summing up McCabe is done by taking away 1 from each Entry summed. This is because a callchain of Entrys without any condition (no IF, CASE, WHILE etc) has only one way of execution and hence the complexity 1. With one IF-statement in the whole callchain it would be 2.

  Useful for**:**
  An overall value of the complexity in the project.
  Can be used to monitor development to avoid complex programming.

- **Missing Entrys.rmq**
  Same as "Distinct Missing Entrys.rmq" but with duplicate entries in the result list. Also show where the entry is being called.
  Note: When trying this on an ASP.NET (C#) project you may get a lot of results, like "(" for the line "tblOverdracht.Border = new BorderInfo((int)BorderSide.All, 1F);".
  This is normal because of the behavior of the C# parser. You can ignore hits that obviously are not an Entry. Be aware that properties are Entrys. The result of this query is normally a long list. It shows all used Entrys and Properties of the class library and 3:rd part libraries. You should not find any of the Entrys/Properties of your own source here, but that is cumbersome to check though. It's better to use the 'Distinct Missing Entrys.rmq' only

  Useful for:
  Identifying entries which are not defined

- **Missing Objects.rmq**
  Same as "Distinct Missing Objects.rmq " but with duplicates in the result list. Also show where the object is found

  Useful for:
  Identifying missing source code

- **Most central Entrys.rmq**
  Overview of the most used entries. Also show the number of dependencies.

  Useful for:
  Identifying the most important entries in code

- **Most central Objects.rmq**
  Overview of most used objects. Also show the number of dependecies.

  Useful for:
  Identifying the most important objects in code.

- **Most controlling Entrys.rmq**
  Central counts dependencies TO + dependencies FROM as dependencies TOTAL. Controlling counts only dependencies FROM.

  Useful for:
  Understanding structure of the most important entries in code.

- **Most controlling Objects.rmq**
  Central counts dependencies TO + dependencies FROM as dependencies TOTAL. Controlling counts only dependencies FROM.

  Useful for:
  Understanding structure of the most important objects in code.

- **Most used Entrys.rmq**
  Central counts dependencies TO + dependencies FROM as dependencies TOTAL. Used counts only dependencies TO.

  Useful for:
  Understanding structure of the most important entries in code.

- **Most used Objects.rmq**
  Central counts dependencies TO + dependencies FROM as dependencies TOTAL. Used counts only dependencies TO.

  Useful for:
  Understanding structure of the most important objects in code.

- **Not used Objects excluding Subobjects.rmq**
  Overview of unused objects.
  Subobjects for .NET and other GUI languages are components on a Form.
  They often have duplicate names, e.g. TextBox1 on several forms.

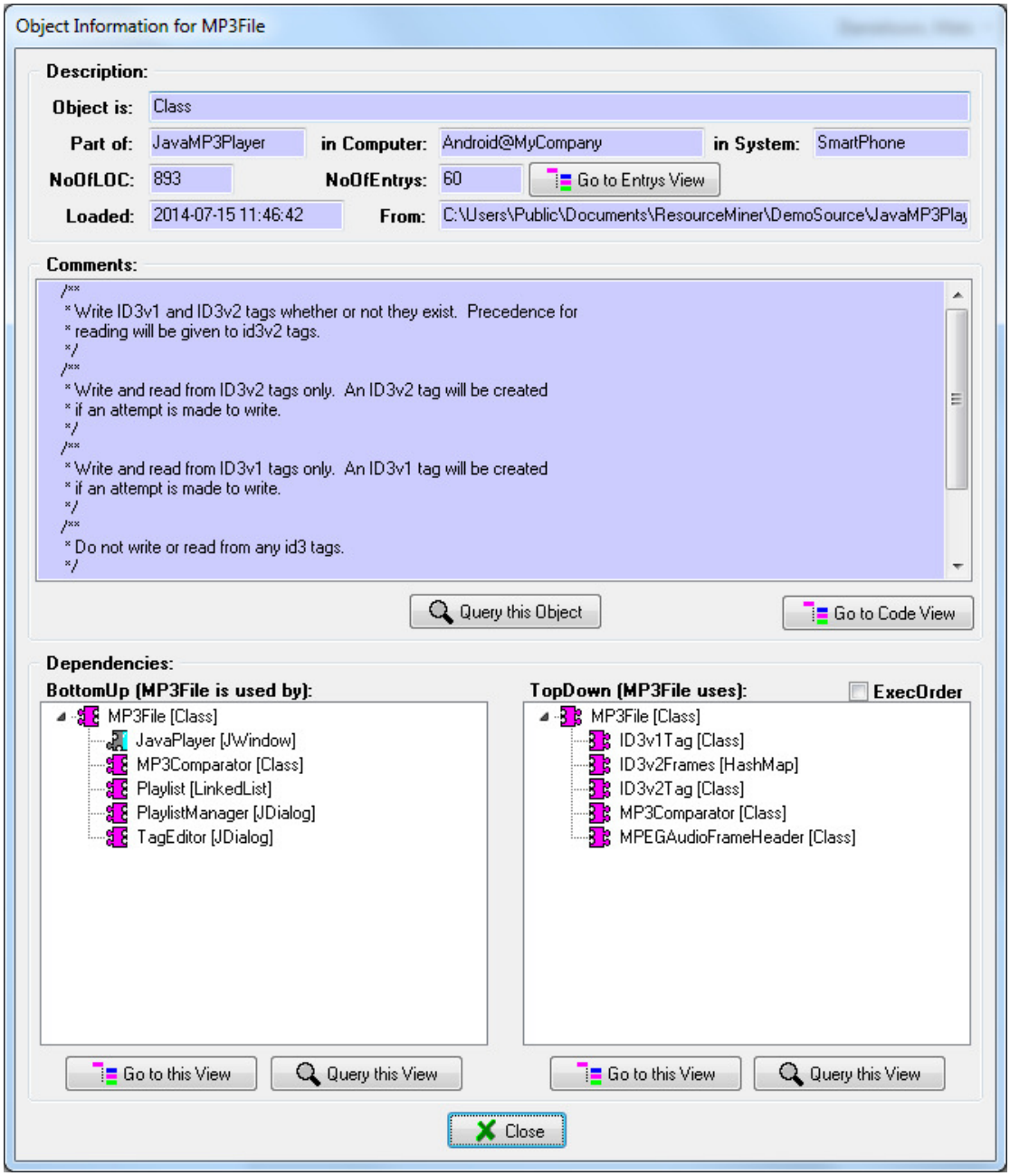
  Useful for:
  Identifying code that can be removed to reduce the complexity.

- **Not used Objects.rmq**
  Overview of unused objects.
  For .NET and other GUI languages, use the script above (excluding subobjects) to get a clear answer. This report (including subobjects) is good for procedural languages without GUI, eg Mainframe Cobol.

  Useful for:
  Identifying code that can be removed to reduce the complexity

ObjectInformation

This form shows an overview about a chosen object:



It is possible to navigate the dependent objects by double-clicking them. It is easier to use then navigating View- and Queryforms, but not as detailed.

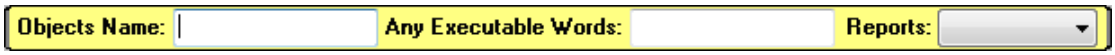
A popupmenu in the 'mini'-views give a few more functions.

ObjectInformation has a number of buttons to get directly to the View- and Queryforms with preset settings.

Wizard

Wizard is a little always-on-top window that makes ResourceMiner very easy to access.

To activate the Wizard, see **Settings – Other** below. ResourceMiner needs to be restarted after changing the settings to Wizard = Yes.



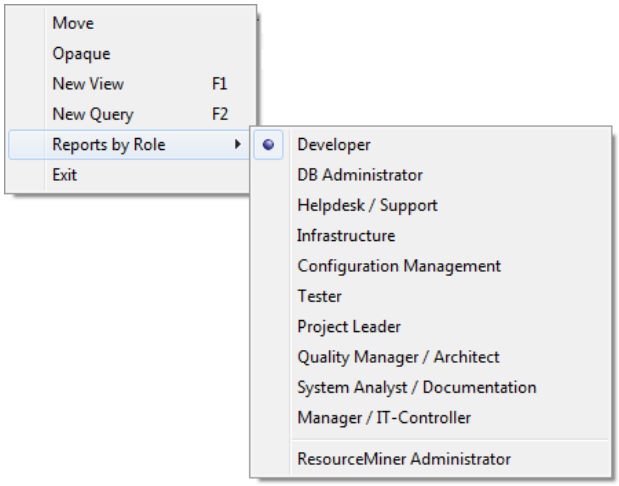
Typing *Objects Name* and <Enter> will show the object in **ObjectInformation**.

Typing *Any Executable Words* and <Enter> will show result in **Query for Statements**.

Typing both *Objects Name* and *Any Executable Words* and <Enter> will make **View** expand the object and also query the expanded Track for the words, showing the result in both **View** and **Query for Statements**.

Predefined, often used, *Reports* can be available from the Wizard.

Right-click on the yellow surface to see the popupmenu:



The popupmenu makes it possible to move and pin the wizard on the desktop, make it half transparent and go directly to a New View or a New Query.

Reports can be published through the Wizard by putting them in the folders under COMMON_DOCUMENTS\ResourceMiner\ReportsByRole. The role *Developer* is default.

There are three kinds of reports to use:

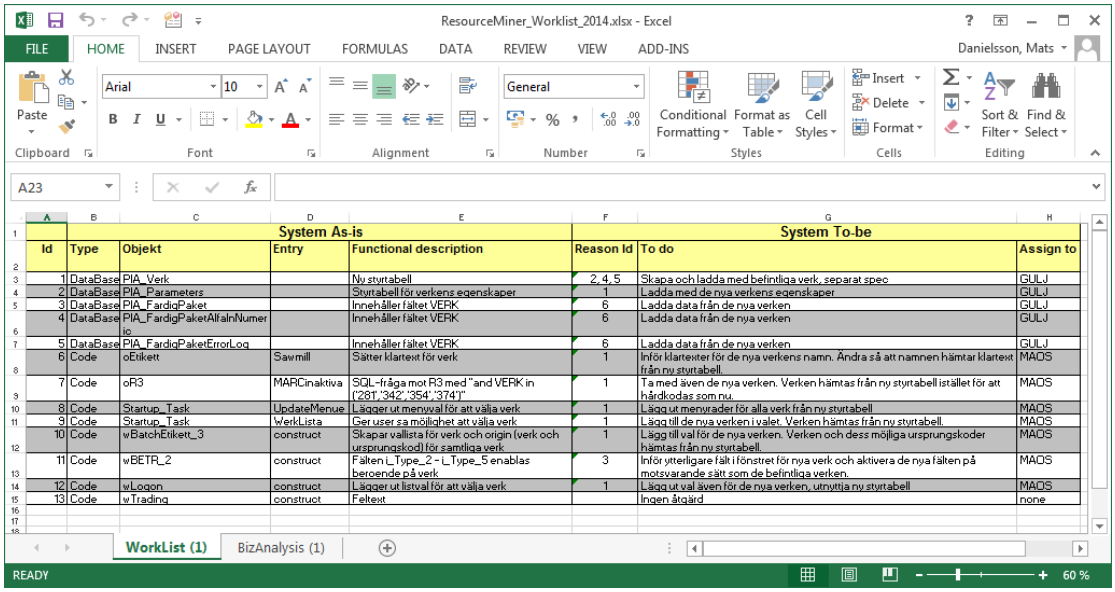
1. View-forms can be saved as *.rmv-files
2. Query-forms can be saved as *.rmq-files
3. ExcelReports using VBA-macro referencing the RMLIB.dll (see OnTopApps below)

Tip: Make shortcuts in COMMON_DOCUMENTS\ReportsByRole\Developer
pointing to your favorite reports in COMMON_DOCUMENTS\ExcelReports
and other places.

Develop

This extension contains functions to create a WorkList and BizAnalysis, support for integration with different development environments (ie Mainframe, Unix, Repositories and Editors) and gives possibilities to define static or dynamic connections (middleware) between environments within the source code.

WorkList



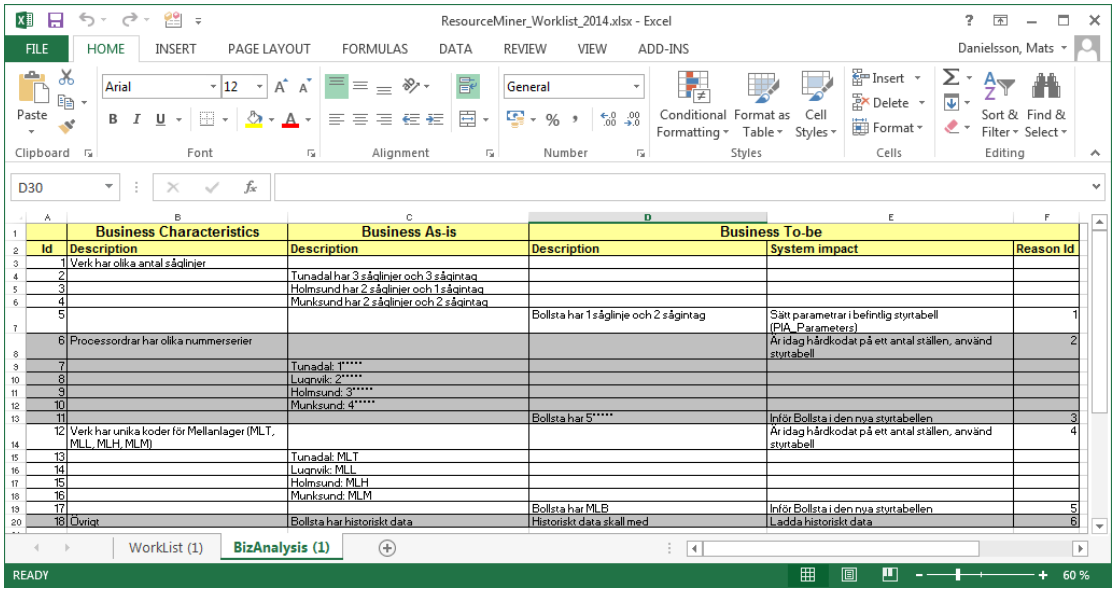
WorkList is an Exceldocument used for adding information from within ResourceMiner about things to change in the analyzed application. It is also possible to fill out WorkList manually in Excel. A new WorkList is created by choosing *File/New WorkList* in ResourceMiner mainmenu.

The button *Open selected WorkListRow in Editor* in WorkList is starting a VBA-macro which starts the proper application and place the cursor on the given row. It is not necessary to have ResourceMiner installed to use WorkList to open source code files. The VBA-macro requires that the row was added by ResourceMiner though.

It is possible that one user that has ResourceMiner creates the WorkList, perform the analysis and then distribute the WorkList to colleagues that has Excel but not ResourceMiner. The colleagues can implement the changes by clicking *Open selected WorkListRow in Editor* and type the changes in their developer environment.

When adding rows to WorkList from ResourceMiner, some controls are made that the column headers match and not have been changed and then the row is added at next empty ID-field (the leftmost column). The Exceldocument therefore must have the column headers unchanged. It is possible to add user defined columns to the right of the AssignTo-column.

BizAnalysis



	Business Characteristics		Business As-is	Business To-be		
	Id	Description	Description	Description	System impact	Reason Id
3	1	Verk har olika antal såglinjer				
4	2		Tunadal har 3 såglinjer och 3 sågintag			
5	3		Holmsund har 2 såglinjer och 1 sågintag			
6	4		Munksund har 2 såglinjer och 2 sågintag			
7	5			Bollsta har 1 såglinje och 2 sågintag	Sätt parametrar i befintlig styrtabell (PIA_Parameters)	1
8	6	Processordrar har olika nummerserier			Är idag hårdkodat på ett antal ställen, använd styrtabell	2
9	7		Tunadal: 1*****			
10	8		Luqnvik: 2*****			
11	9		Holmsund: 3*****			
12	10		Munksund: 4*****			
13	11			Bollsta har 5*****	Inför Bollsta i den nya styrtabellen	3
14	12	Verk har unika koder för Mellanlager (MLT, MLL, MLH, MLM)			Är idag hårdkodat på ett antal ställen, använd styrtabell	4
15	13		Tunadal: MLT			
16	14		Luqnvik: MLL			
17	15		Holmsund: MLH			
18	16		Munksund: MLM			
19	17			Bollsta har MLB	Inför Bollsta i den nya styrtabellen	5
20	18	Övrigt	Bollsta har historiskt data	Historiskt data skall med	Ladda historiskt data	6

BizAnalysis is an Exceldocument that the user fills out manually, it is not possible to add Items or Line of Code from within ResourceMiner. BizAnalysis is an extra sheet existing in all WorkLists, it is hidden but can be unhidden by ResourceMiner by choosing *File/New BizAnalysis* in ResourceMiner mainmenu.

BizAnalysis is used to describe characteristics of a business that has any kind of meaning for the application change to be done in WorkList. Some business characteristics of interest can be known from start, other can be discovered bit by bit as the source analysis goes on.

The mapping between WorkList (what to be changed) and business characteristics is done in the column ReasonID. ReasonID also exists in WorkList. When doing *Add to WorkList as* from within ResourceMiner existing Reasons in BizAnalysis can be selected to connect Reasons or add new Reasons.

Business rules are often implemented in several places in the source, which means that one changed business rule gives several rows in WorkList. By shortly describing the business characteristics of interest mapped to what source changes to make, find those places in ResourceMiner and map them by ReasonID gets a good tracability and control over the analysis.

BizAnalysis can also – after completion – be used as a simple but yet quite detailed requirement and testcase document to perform the necessary tests after implementation of the changes.

When ResourceMiner add rows to WorkList, some controls are made that the column headers in BizAnalysis match and not have been changed before showing existing Reasons. The Exceldocument therefore must have the column headers unchanged. It is possible to add user defined columns to the right of the ReasonID-column. Reasons will only show up in ResourceMiner when ReasonID has a number and Business To-be Description or Business To-be System impact has text on the same row.

Integration with developer environments

Develop contains extended support to integrate ResourceMiner with different environments where source code is being handled:

- Possibility to run scripts that automatically get wanted versions of source code files from IDE, version control system, other environments than window etc. The scripts run before Load.

  Example script for Visual SourceSafe comes with the installation, its name is **vss.bat** and contains comments on what it does. The section *Load* below describes how to make the script execute before each load. Own scripts can be written for source files FTP from Unix or Mainframe etc.

- Possibility to make ResourceMiner open and show Items/Lines of code in IDEs/Editors. The section *Settings – Editors* below describes how to make your favorite editor to open and position itself on what been found in ResourceMiner.

- Possibility to make WorkList open and show Items/Lines of code in IDEs/Editors without having ResourceMiner installed. See the section about WorkList above. The settings in *Settings – Editors* is also valid for WorkList.

Define middleware

Systems of today are often written in several different languages and environments connected by different types of third party software (middleware). For example webpages or fat clients accessing stored procedures in databases or old mainframe programs. Dynamic dependencies where a string contains the dependency are also common in modern programming languages.

To be able to follow callchains and get the complete picture of dependencies within and/or between systems one needs to find and define the missing links. The missing links can be found using ResourceMiner.

The section *Settings – Middleware* below describes how links can be defined and tested to make ResourceMiner show all dependencies in the code.

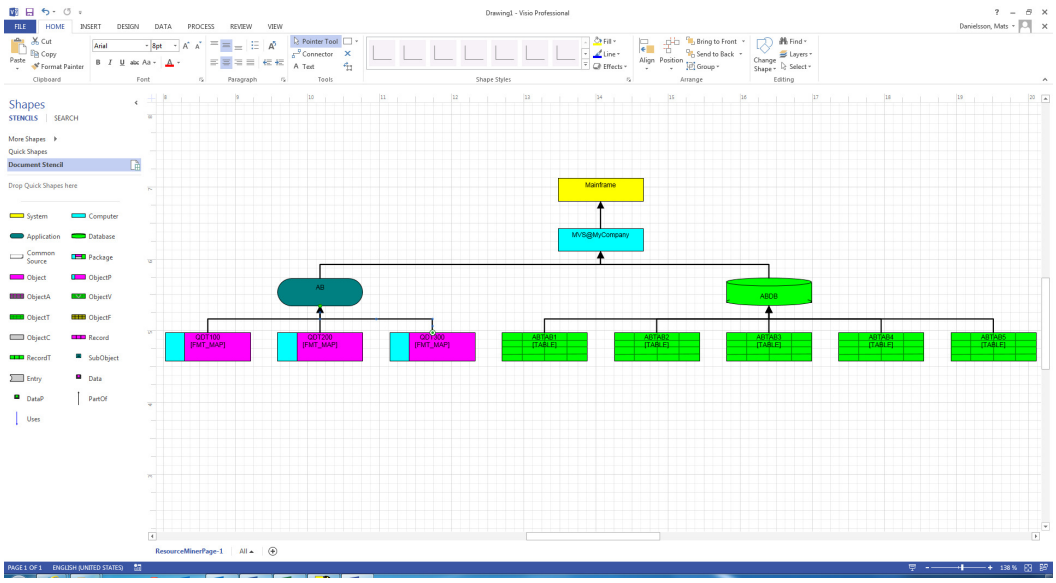
Document

This extension contains integration with Visio to generate diagrams, a Measure functionality and extensions to automate analysis by macro coding in Excel, named OnTopApps.

Visio

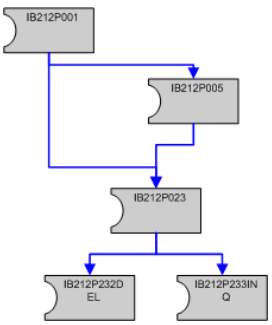
ResourceMiner can generate diagrams (boxes, lines and text) to Visio which can be customized to different types of drawing standards.

A new Diagram is created by choosing *File/New Diagram* in ResourceMiner mainmenu. The Diagram inherits its properties from a template containing ResourceMiner shapes (stilistics of the Icons in the View-form). The shapes can be changed as pleased, but the name of the shapes must not be changed if ResourceMiner should be able to do its job properly.



Diagrams are built up by adding Tracks from within ResourceMiner. It is also possible to work manually on the diagrams in Visio.

Tracks can be added on top of each other so that for example many-to-many relationships will show:

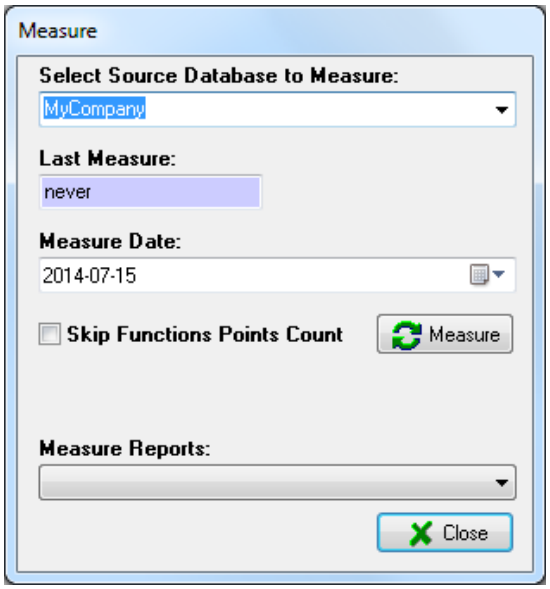


When adding Tracks to a diagram, each Item will only show up one time. Relationships on the other hand will be added as more and more lines.

The diagram can be customized and extended with texts and other explanations, rearranged, printed and saved with Visios standard functions.

Measure

This function is used to perform automated measures of size, complexity and quality of application source code, the results are saved in the MeasureDB and the history (the trends) can be studied by managers. APM = Application Portfolio Management.



As default measures will go into RM_FirstMeasureDB.mdb. The database used is defined by a connection string set in the Other Settings (see above) at 'Measure database for APM'.

Functions Point Count presupposes that a function-filter has been set in Settings/Functions (see above).

Click the 'Measure'-button to start counting. The progress is logged to MeasureLog.txt (at same location as LoadLog.txt).

The results inside MeasureDB can be extracted to an Excel spreadsheet. A predefined measure report AllMeasureDataRaw.xls is included. APM dashboards with graphical trends can be developed with Excel.

Measures can be started from Windows command line or by bat-files from scheduled task:
ResourceMiner.exe MEASURE_ON_DB_NAMED <dbname> <SkipFPCount> <ForceDelete> <MeasureDate>

Example 1, today and include Function Points Count:
ResourceMiner.exe MEASURE_ON_DB_NAMED RM TutorialDB No

Example 2, force a date and skip Function Points Count:
ResourceMiner.exe MEASURE_ON_DB_NAMED RMTutorialDB Yes Yes 2009-10-25

OnTopApps

By clicking in RM any kind of analysis should be possible to do. Some analysis however takes a lot of clicking and would be nice to automate as reports.

RMLIB.dll is a COM DLL for Windows that can be used by Excel VBA-coding or any other macro/program that needs to use the information from RM databases. The COM DLL can be consumed from .Net by wrapping techniques.

Excel with macros is the most common and compatible format without having to install and adjust any underlying framework. A number of reports based on Excel VBA-macros, the RMLIB.dll and rmq-reports are available for use in the folder COMMON_DOCUMENTS\ResourceMiner\ExcelReports.

Instructions for using the Excel reports:

1. Open any of the Excels in the ExcelReports folder.
2. Open the Visual Basic Editor (might need to give a password to see the VBA-source)
3. Make sure that VBA Tools/References includes RMLIB.dll
4. Read the instructions on the Settings-sheet, fill in your RM-database (as named inside RM) and fill in the name of an Object in that database.
5. Click the Start-button
6. See result in Current_Result sheet. Should match with what you can see in the View-form of RM.

With each Excel comes some .rmq-files. Those can be opened, edited, saved and created in RM (it is Query SQL Reports).

All reports can be further developed with new SQL-statements and logic.

Example 1:
Harvesting Business Rules takes a lot of repetitive clicking in RM.
Adjust and enhance reports to automatically present the rules as wanted.

Example 2:
For a renewal strategy it might be needed to define the exact scope of an application. The starting-points are known (forms and batches) but what will the interfaces be when this application is moved to another platform? By running TopDown- and BottomUp-reports the task is much easier.

Example 3:
Generating new source from the model inside an RM-database.
Language conversion, web services or data handling routines are just a few examples of what can be generated from the source model inside the RM database.

To make use of those powerful features you need:
- Figure out how to navigate (click) in RM to get your answer, or part of answers
- Understand the data model of the RM-database to be able to write the SQL-statements needed
- Adjust the VBA logic in one of the Excel reports to get the output needed
- Test and verify the output

The time to develop solutions on top of RM can be justified by time saved by automation and/or easy use for infrequent users, for example as a FAQ on source code, living documentation.

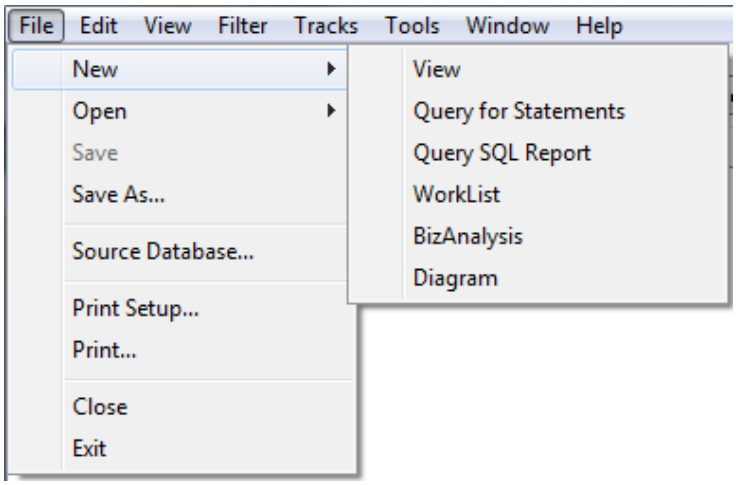
The reports can be published through the Wizard (see above) by putting them in the folders COMMON_DOCUMENTS\ResourceMiner\ReportsByRole.

Using RMLIB involves license checking for accessing RM databases. RMLIB can also be used for publishing services (reports) from RM databases to any user if floating licenses is used.

Common mainmenu

The mainmenu is common to all View- and Queryforms in ResourceMiner, the common functions are:

File



New View
Opens a new Viewform, makes it possible to alternate between different views.

New Query for Statements
Opens a new Queryform with the tab *Query for Statements* active, makes it possible to alternate between different database queries.

New Query SQL Report
Opens a new Queryform with the tab *Query SQL Report* active, makes it possible to alternate between different database queries.

New WorkList
This choice is part of the extension *Develop*. If Develop is installed, a new empty WorkList will be opened in Excel.

New BizAnalysis
This choice is part of the extension *Develop*. If Develop is installed, a new empty BizAnalysis will be opened in Excel.

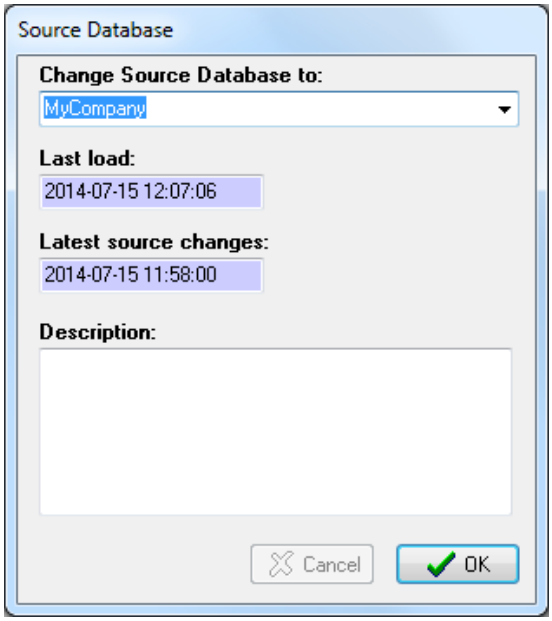
New Diagram
This choice is part of the extension *Document*. If Document is installed, a new empty Diagram will be opened in Visio.

Open
Open has the same subchoices as New.
The user selects the file to be opened. Can be View (.rmv), Query (.rmq), WorkList (.xls), BizAnalysis (.xls) or Diagram (.vsd). The opened document will be shown in current or in a new form.

Save
Saves current form as a document, if the document has not been saved before this choice is disabled, user should use Save As instead.

Save As
The user give a name to the document and selects where the document will be saved.

Source Database



Source Database gives the possibility to connect the current form to another source database and will show when source databases was loaded last time. Connection to source database is inherited from current form into new forms when doing 'New…' or 'Show in…' functions. The database connection is stored with a saved document.

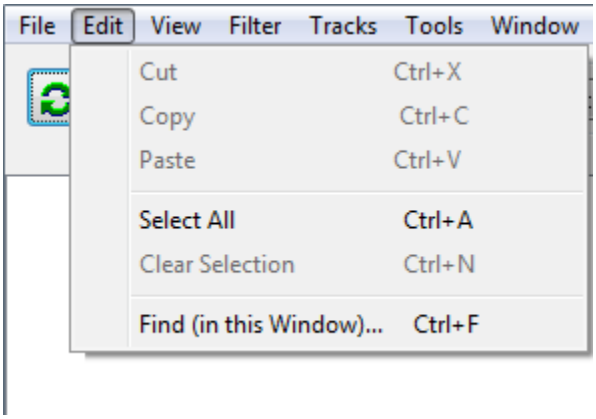
Print Setup
Sets the printer information for current form.

Print
Prints current form.

Close
Closes current form.

Exit
Ends the application, all forms closes.

Edit



Cut
This choice is enabled when focus is on editable boxes, clear selected text and saves it to windows clipboard.

Copy
Copy selected text and save it to windows clipboard.

Paste
This choice is enabled when focus is on editable boxes, paste text from windows clipboard to current editable box.

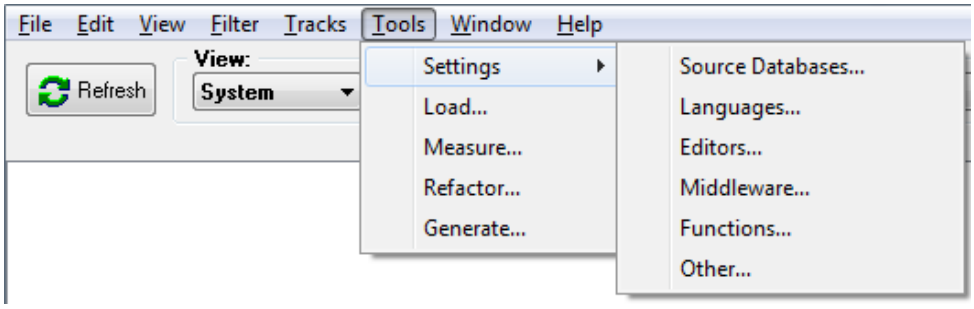
Select All
Select all text/all items in current form or focused section of a form.

Clear Selection
Deselect all text/all items in current form or focused section of form.

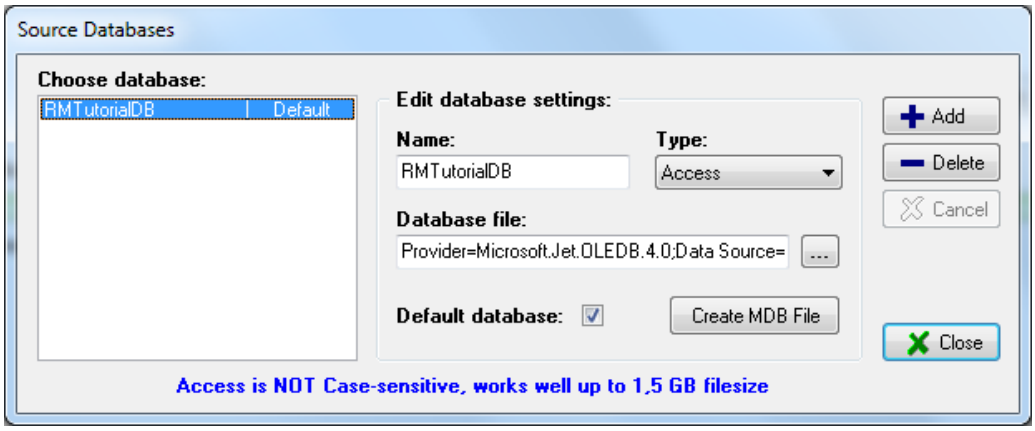
Find (in this Window)
User gives a text to search for, search is performed in current form or focused section of form. If the text is found, the form is scrolled so that the found text shows and the found text is highlighted.

Observe that the search starting point is from selected/highlighted text if such exists, otherwise the search starting point is at the beginning of current form or focused section of form.

Tools



Settings – Source Databases



ResourceMiner is aware of databases through this form.

The list contains the known databases, only one database can be *Default*, ie be the database that is connected when ResourceMiner start (if starting ResourceMiner by opening a rmv- or rmq-document, the document itself contains what database to connect to, which can be another database than the default).

The button *Add* adds a database to the list:

1. Set database name, can be any name that says something about it's content, for example the system name or the company name if the source database contains all or several systems for a company.
2. Choose database type. Read the blue text which change with selected database type.
3. Depending on database type, create the dbfile by a click on the 'Create'-button or set the connection string.
4. Save and Close.

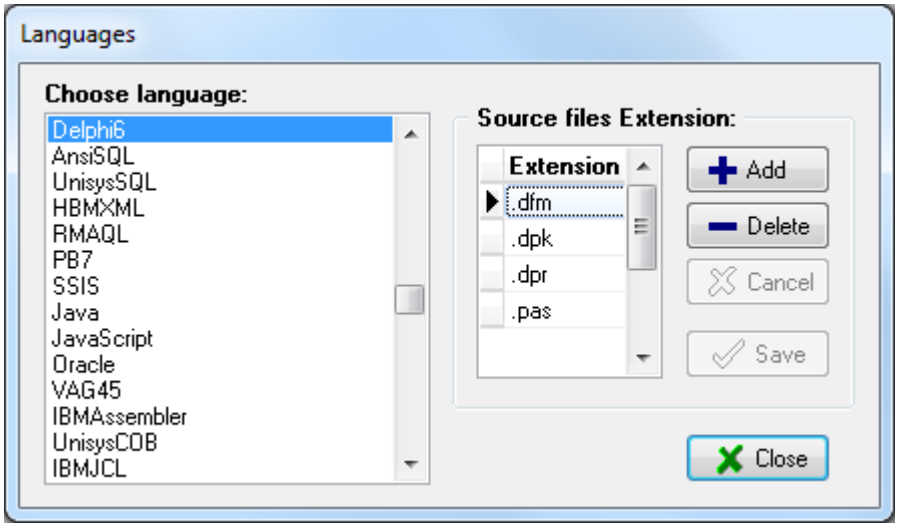
Tip: Do not set the database to default until tested that the database connection is working. Test connect by *File/Source Databases*.

The button *Delete* removes a database from the list (the physical database is not removed, only ResourceMiner knowledge about it is affected).

Good to know about database connections:

- A saved document 'knows' which database it should connect to when opened.
- When choosing **File\New…** or **Show in…** the database connection is inherited from current form.

Settings – Languages

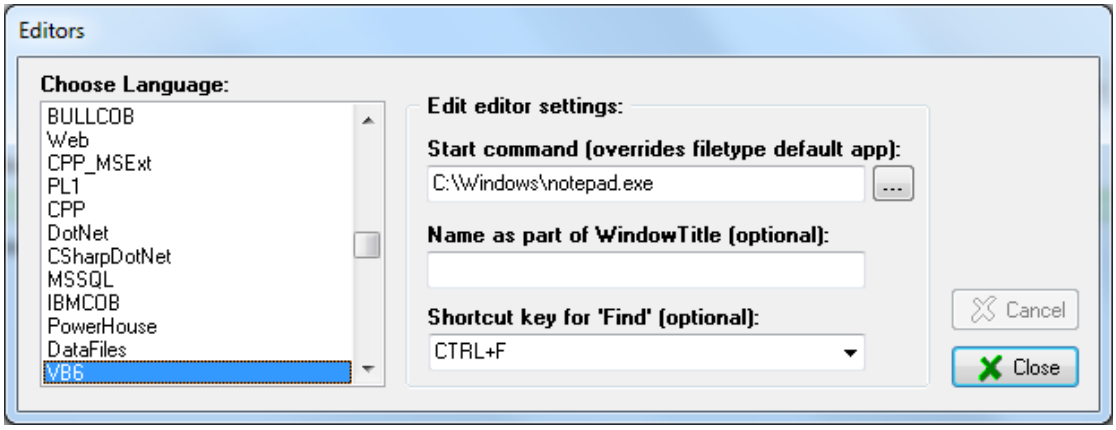


The leftmost list show which language definition extensions ResourceMiner can use. The list depends on language extensions installed and ResourceMiner license level.

As default each language definition work on files with the file name extensions in the rightmost list. If your sourcefiles have other file name extensions than these defaults, it is possible to change, add and delete what file name extensions the language definition should work on.

Settings – Editors

This function is part of the extension *Develop*.



When using the choice *Show in Editor* in View- or Queryform, the source file is opened with the application which is linked to the filetype in Windows.

In this form it is possible to override windows default. The settings is unique for each language.

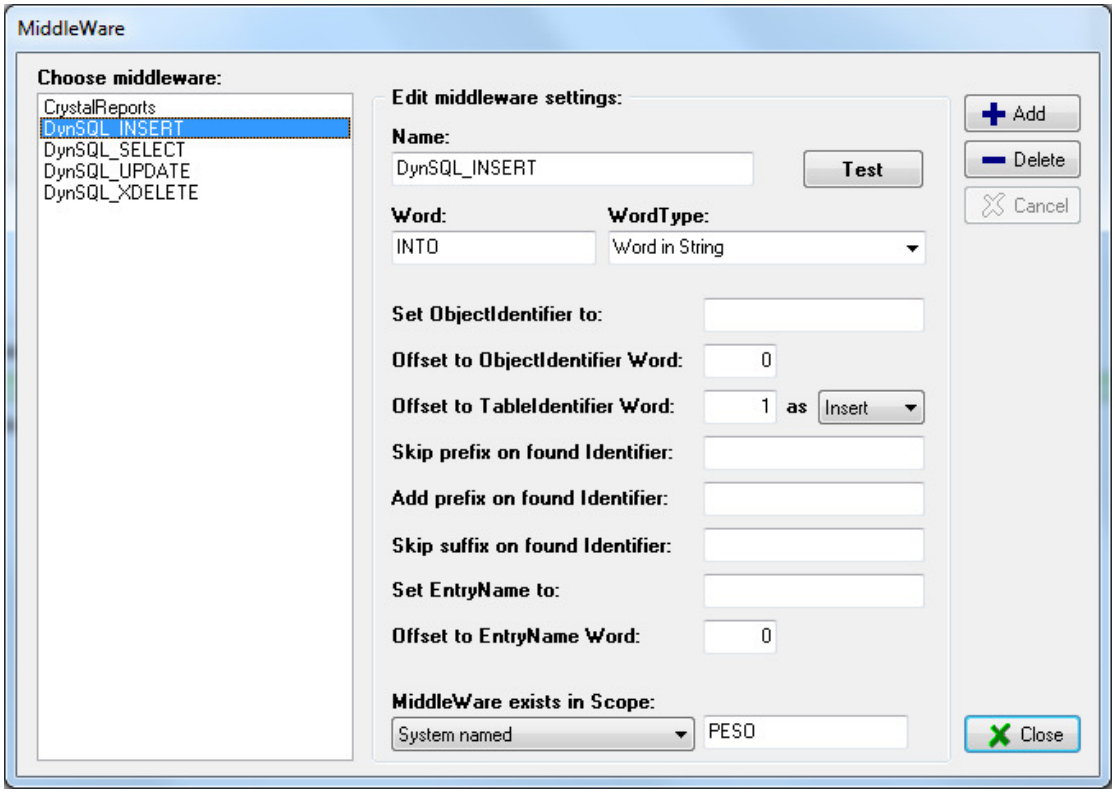
Start command
Alternative executable file that takes path and filename as its first argument. Can be used to perform checkout from a repository and then open the editor.

Name as part of WindowTitle
When the source file has been opened in the Editor, the window containing the source code should be focused. ResourceMiner try as default to focus the window that contains the file name in its title. If your IDE/Editor not focus properly, it is possible to make it focused by setting a text that the source window has in its title.

Shortcut key for 'Find'
When the source is opened and focused, it is possible to make your editor to search for the line of code to show. Set the keyboard sequence you would hit to start the search/find function of your editor. These keys can be used: CTRL, SHIFT, HOME, ESCAPE, A – Z and 0 – 9.

Settings – Middleware

This function is part of the extension *Develop*.



ResourceMiner find and resolve all static dependencies within the source code, ie includes, calls, pointers etc.

There is also another type of dependencies one should be aware about; dynamic dependencies where a call and it's parameters contains the dependency, normally in a string parameter. This is quite common in modern programming languages but can exist in mainframe languages as well. The called function or program that takes the parameter is a middleware. Middlewares are often 3:rd party products or components and its source code is not available or part of the own source. A middleware makes a call or connection to the System/Subsystem/Object/Entry given as parameters.

To be able to get a correct picture of the applications inner structure within and/or between systems one needs to find and define the missing middleware links. The missing links can be found using ResourceMiner to see what Objects and Entrys is not used at all and then search the source for their names. If the name exists in the code, verify if used as parameters to middleware functions of any kind.

MiddleWare settings are as default stored in a separate csv-file. See Settings – Other below.

For each pattern of dynamic dependencies found, define the linkpattern in the middleware form:

Name
Name on the Middleware, can be any name that says something about the type of linkpattern.

Word and WordType
Type a word that will give hits in the source on the middleware function or keyword before the word containing Objectname or Entryname to link. Example:

```
//S020     EXEC PGM=ADUUMAIN,REGION=0M,
//         PARM='DB2M,STYDEB02,NEW,,MSGLEVEL(0)'
```

The program ADUUMAIN takes a number of parameters (PARM=…), one of the parameters is the database table (STYDEB02) that this operation (LOAD or UNLOAD) works on.

The word ADUUMAIN of type Call will give the correct hits in the source.

Hardcoded ObjectIdentifier
In some cases the called Entry is given in the parameters but not the target object. It is possible to set a fixed name for the target Object.

Offset to ObjectIdentifier, TableIdentifier or EntryName
Set a number for how many words forward from the given keyword the ObjectIdentifier, TableIdentifier or EntryName resides. In the ADUUMAIN example above it is a TableIdentifier residing 11 words after the keyword.

For TableIdentifier it is possible to set type of access; Insert, Select etc.

Skip prefix on found Identifier
For situations where the found Identifier must be altered to match the target Object.

Add prefix on found Identifier
For situations where the found Identifier must be altered to match the target Object.

Skip suffix on found Identifier
For situations where the found Identifier must be altered to match the target Object.

Hardcoded EntryName
For situations where the called Object is given in the parameters but not the target Entry. It is possible to set a fixed name for the target Entry.

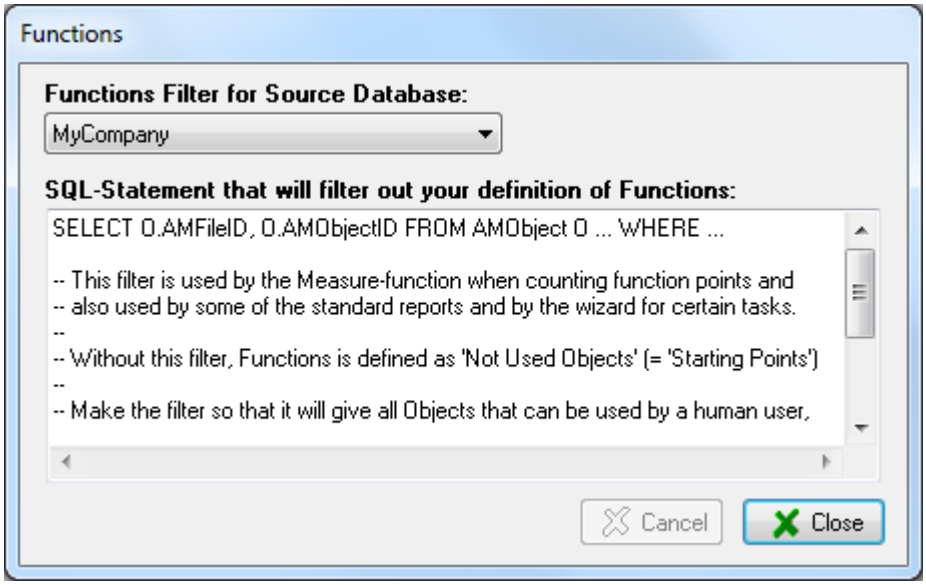
MiddleWare exists in Scope
Select in which scope this middleware exists.

The button *Test* will show the hits on keywords in the source and also what ObjectIdentifier, EntryName or TableIdentifier found with the given offsets.

Defined middlewares are used when ResourceMiner Loadfunction (see below) perform the DependencyChecks. Changed middlewares settings is not affecting dependencies until the DependencyChecks has been performed again.

Settings – Functions

This setting is stored in each source database in the AMAttribute table. It is used by the Measure function (see below) when counting Function Points.



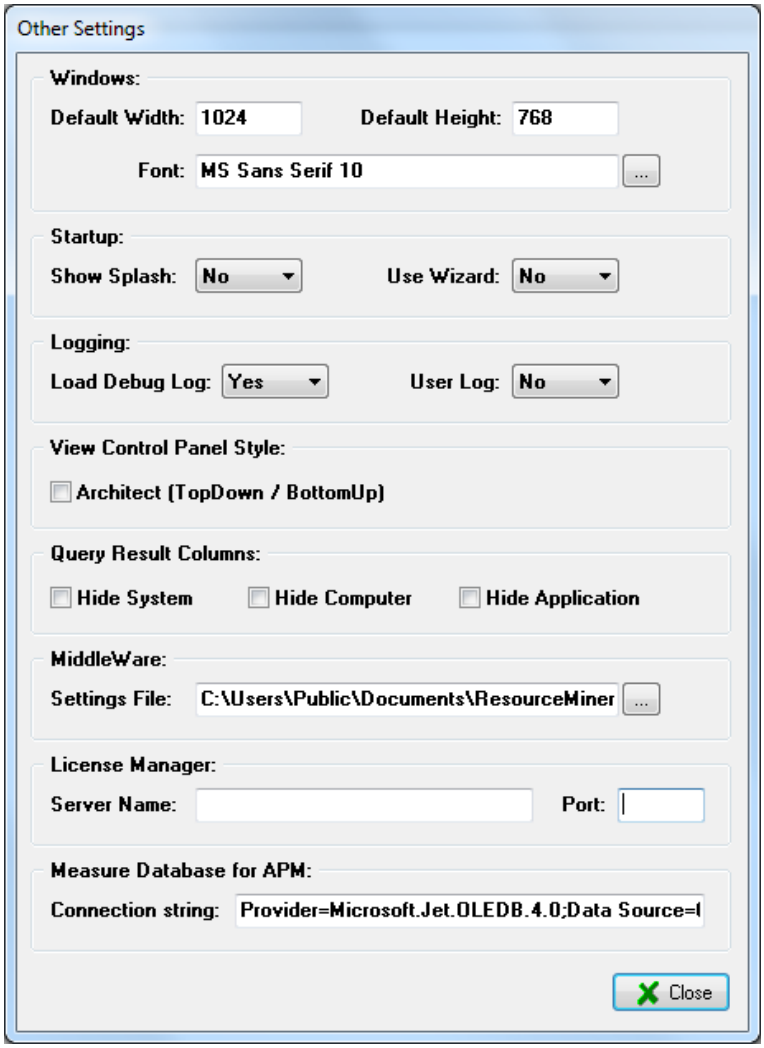
The SELECT-statement shall give all the starting points in the source database.
Without this filter, Functions (starting points) is defined as 'Not Used Objects'.

Make the filter so that it will give all Objects that can be used by a human user, for example Online transactions, GUIs, Webpages. Also include batches and scripts that can be started by a user / operator. Also include objects that acts as APIs for other external applications.

Tip: Filter on naming standards on ObjectName and/or its inheritance

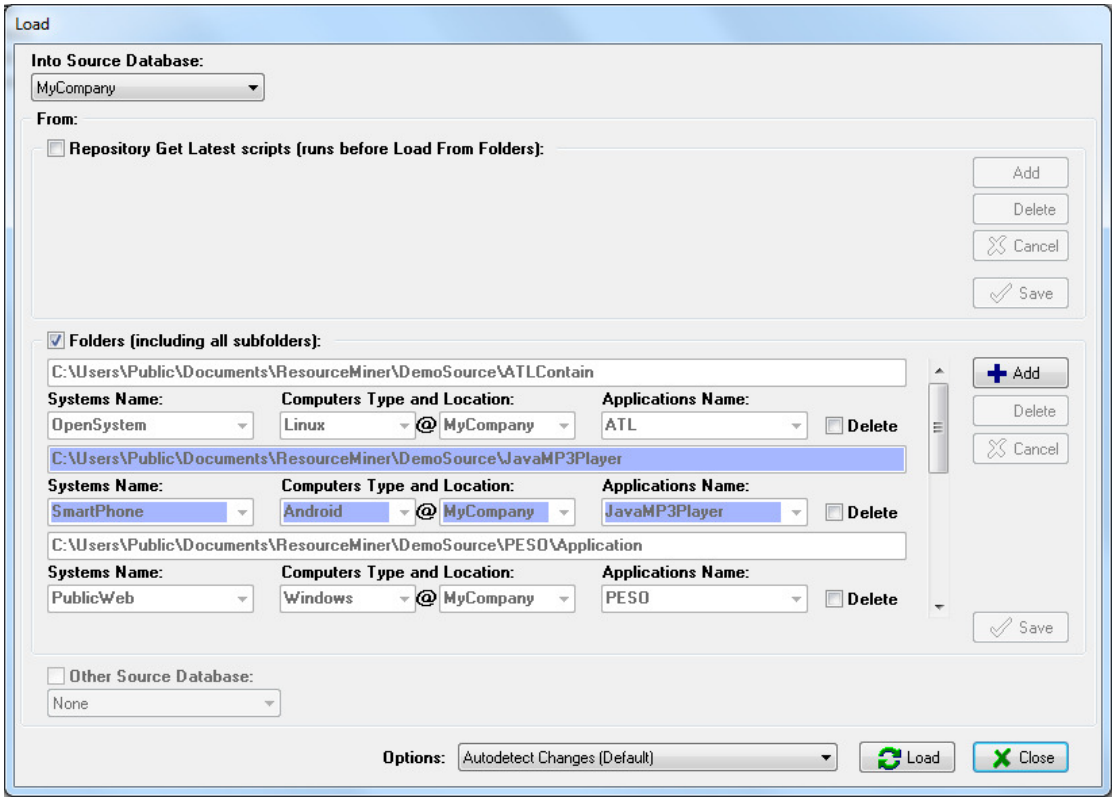
Settings – Other

Those settings are read from and stored in RM.ini:



Restart ResourceMiner for changes in the ini-file to take effect.

Load



This function loads source code from one or several folders (including subfolders) into the chosen database. Before loading from folders, it is possible to run scripts that automatically get wanted versions of source code files from IDE, version control system, other environments than windows etc to the given folders.

Load from *Other Source Database* will synchronize the chosen database to get its content identical with the other database. This is an extra feature that requires a special extension.

The load information given in this form is saved in the database it belongs to and need not be changed in order to reload the source. Load does the following:

- Given Repository-scripts executes
- Given folders and all its subfolders (recursive) is being searched for all file extensions that are given for all languages in the Language settings list. Matching files are parsed into the database.
- All middleware settings are prepared to be resolved by dependencies
- All middleware and static dependencies in the source code is identified and saved to the database.

First (initial) load will parse all matching source files in the folders and its subfolders (recursive). Secondary loads will parse matching files that have been changed or added since last load (looking at changes in date/time/size). Secondary loads will also remove information in the database about files that have been deleted since last load.

A loaded source file that have been moved from one folder to another will at next load be treated as deleted and as a new added source file.

During load a progress window shows what is happening, progress information is also logged to the file **LoadLog.txt** in the Logs-folder of COMMON_DOCUMENTS\ResourceMiner.

Load instruction:

1. Make sure the source files are available as readable text files in a folder or tree of folders. If the source files resides in a version control system (MS SourceSafe, PVCS etc), get them out by 'Get Latest' or a simular function. To avoid the manual 'Get'-procedure it is possible to write a script for the task and make ResourceMiner run the script before loading from folders.
2. It is recommended that the source files are organized in one folder or tree of folders per application. Source files that is common for several applications can be placed in a common folder at same level as the applications folders.
3. If the application(s) uses one or more databases, the DDL-scripts that describes the databases (a readable textfile with commands like CREATE TABLE xxx…) should be used as SQL source files. The DDL-scripts can in most databases be generated from the database management console. It is recommended to place DDL-scripts in one folder per database at same level as the applications folders.
4. Choose Tools/Settings/Languages at the mainmenu and check that the source files is matching the file extensions given for the language definitions of interest. Change the file extension settings if needed.
5. Choose Tools/Load at the mainmenu.
6. Choose **'Into Database'** to load the source into.
7. IF the loaded source files is located in another user's PC, be aware that the **'From Folder'** can point to folders not existing or not reachable from your PC.
8. IF there is need to change a **'From Folder'**, it must be done with Delete and Add, which also will empty parts of the database. This can take some time.
9. Add one or several folders as **'From Folder'**, choose Add and point out the folders like this:
   a) Add one Folder per application (if the files are organized in that way)
   b) Common source for several applications should be loaded from its own Folder
   c) For each Folder, the fields Systems, Computers and Applications Name should be given:

System:	The systems name, if several Folders is part of the same system, be careful to give same name (casesensitive) to all these Folder rows.
Computer:	Type of computer, for example Client, Server, Mainframe etc. The @-sign means where the computer is physically placed, for example Client@SCA,WorkStation@Sogeti. If several Folders is part of the same computer, be careful to give same computer type@location (casesensitive) to all these Folder rows. Observe! If the folder contains common source for several computers, for example both client and server, then leave this and next field blank.
Application:	The application, subsystem or database name. Observe! If the folder contains common source for several applications, then leave this field blank.

10. Click the Load button. Loading the database can take some time depending on the amount of source code and the performance of the computer. One might want to load during lunch or over night. It is possible to run other applications concurrently but the computer will probably be kind of lacy.
11. IF the load is interrupted, for example by shutting down the computer or killing the application in task manager, it is possible to restart the load later on. ResourceMiner know where it stopped and will start from that point next time Load is started.
12. When done, start with checking the quality of the loaded source and check the need for middleware settings. See the Users Guide for how to go on from here.

Secondary loads (keeping the database up to date with the source files) only involves step 5 and 10 in this instruction. Only changes will be handled, the load time will be much shorter.

For secondary loads it is possible to load automatically by Scheduled Task, see example script **BackgroundLoad_RMDBs.bat** that comes with the installation and contains comments on what it does.

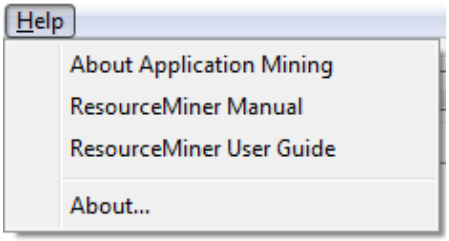
Measure
This function is part of the extension *Document,* see description above.

Refactor
This function is part of the extension *Refactor*. If Refactor is installed it will start the Refactor function.
Refactor is not described in this manual.

Generate
This function is part of the extension *Refactor*.
Refactor is not described in this manual.

Window
Windows can be tiled. This menu also show a list of all open windows (forms) with their names. Choose which to switch to. It is also possible to switch windows (forms) by the keystroke CTRL+TAB.

Help

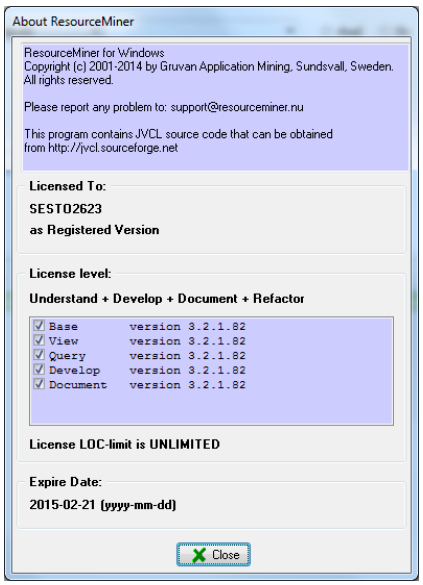


About Application Mining
Opens a webpage (URL) about the concept of Application Mining. Internet connection is needed.

ResourceMiner Manual
Opens this document.

ResourceMiner User Guide
Opens a document describing common use cases in ResourceMiner.

About



Information about your license, components and versions.