1. CONTENTS

2. RELEASE NOTES

2.1. RELEASE NOTES – 9.5.104

This release exists to allow interoperability between 9.4.6/7 versions of the client with the new 9.5.x server and vice versa. There are also fixes for problems with agr_getbrowser.

A full list of changes can be found below.

2.1.1. Changes

Common

1. Legacy features of the IDS implementation have been added back so as to allow use of the 9.4.6/7 clients with the 9.5.x server and vice versa.
2. Problem with agr_getbrowser where the browser template contains “not empty” constraints is resolved.
3. Problem with agr_getbrowser with the treatment of the client / company parameter in browser templates is resolved.
4. Added “location” when selecting Information Browser Report name from list. (Because it is possible to have more than one report with the same name)
5. The layout of the “default [.pdf]” output when using “Export” from Browser Templates and Information Browser Reports is improved to include parameters and page numbers.

Excelerator

6. “declare” is now used rather than “setdefault” when generating template _control and postback sheets.
7. Added “Planner” as a valid entry when building posback sheets.

XtraReports

8. File “DevExpress.Pdf.v20.1.Drawing.dll” is now correctly added to XtraReports runtime files under bin\ReportEngine\Xtra.20.1. Previously this was missing and caused a runtime error when running a report containing the new PdfContent control on the server.

2.2. RELEASE NOTES – 9.5.100

The most important change in this release (and the reason for the updated 2nd digit of the version) is that the version of XtraReports has been upgraded from 17.2 to 20.1. This contains some important improvements notably the new PdfContent control which enables multi-page pdf files (extracted from the document archive or elsewhere) to be appended into the report output file.

Also, change #7 below should be regarded as critical for Excelerator users who have already upgraded to 9.4.7.

A full list of changes can be found below.

2.2.1. Changes

Common

9. There is a new “agr_getDocuments” macro available in ReSQL which is exactly the same as the existing “agr_getDocument” macro except that it returns one row for each matching document rather than just the first matching document.
10. On the Agrxlcmd command line data supplied via the /data:my_data.xml switch will always take precedence over any data that may already exist in an rerx file.
11. Further improvements to the Web API and IDS mechanism.

Excelerator

12. Resolved issue with ## backing sheets being loaded more than once if referred to by more than one Pivot table. Not only did this waste time but in the case where control information is being removed the second load would fail.
13. Null date parameters no longer cause a problem when creating new Excelerator reports based on Browser Templates. (agr_getbrowser ‘template’, date_le=null)
14. Resolved bug in “query balance” which caused incorrect results when a query referred to a relation on relation on relation without referring to the intermediate relation (introduced in 9.4.4).
15. Resolved problem with the logged in client, language and user_id not being seen as parameters when running Excelerator reports within the Excel Add-in. (introduced in 9.4.7)

XtraReports

16. We now ship both XtraReports v17.2 and v20.1 hence the increased size of the installation.
17. The new (v20.1) XtraReports runtime is always used when running reports. This is always advantageous as the detailed alignment of text within controls is improved.
18. By default, the old (v17.2) designer is still used so that reports that need to be run on machines with older versions of ReportEngine can still be created/edited. However, there is a new “Open with latest designer” option on the right mouse menu in the ReportStudio package explorer which allows you update your .repx files to 20.1

and use the v20.1 designer to take advantage of enhanced functionality. When you do this ReportEngine will automatically make a backup of the whole .rerx file before the .repx files are updated.

2.3. RELEASE NOTES – 9.4.700

This release wraps up some recent changes to coincide with UBW 7.6. The most important changes relate to IDS and to problems with the Excelerator “query agresso” syntax. A complete list of changes can be found below.

2.3.1. Changes

Common

19. The WebAPI / IDS connection now works properly for Multi-Tenant applications.
20. Resolved problem with IDS token expiry timeout.
21. Name of dataset in new resql file now defaults to “Data” rather than “New”. (This looks much more reasonable in the XtraReports designer)
22. Data Relations are now preserved when saving _data.xml to, and reading data back from, xlsx files
23. There are new “Export to xlsx” and “Import from xlsx” options available in the “right mouse” menu on _data.xml and _data0.xml in the Package explorer window in Report Studio. (In conjunction with the above change this makes it very easy to edit the data for test purposes)
24. Resolved problem with getting data from Browser Templates with multiple criteria on the same column. (This problem was introduced in 9.4.607)
25. The treatment of parameters in “agr_getbrowser” is enhanced. Parameters marked as prompting in the “SearchC” screen must be provided, others may be provided if desired. Parameters have names like “period_ge” / “period_le”.

Excelerator

26. The treatment of parameters when using “query agresso” reverts exactly to the pre 9.4.4xx behaviour. Only parameters in the “SearchC” screen marked as prompting are processed and have names like “period_from” / “period_to”. N.B. if you want to set values of non-prompting parameters it is straightforward to convert the report to use “agr_getbrowser …” instead of “query agresso …”
27. Word wrap is now properly applied when saving Excelerator output to pdf.

XtraReports

28. Designer layout information is now cleared from the registry if it is found to be invalid. This will cause the designer to open with the default layout the next time it is invoked. (Previously this situation prevented the designer from opening at all)
29. Parameters with names beginning with “_” (underscore) no longer cause a validation warning if they do not exist in the dataset. This is helpful when it is necessary to define parameters to use only within the report. E.g. within the “filter string” of a subreport.

2.4. RELEASE NOTES – 9.4.607

This release contains a number of maintenance fixes some of which will be important for some users. A complete list of changes can be found below.

2.4.1. Changes

Common

1. Resolved a number of problems with “new” locales like “en-NL” including “`Cannot change CaseSensitive or Locale property.`”
2. Added “Run>Predefined…>Default [.xlsx]” to menu bar in Report Studio.
3. Reporting line numbers of ReSQL errors is improved.
4. Data is now persisted as xml (_data.xml) rather than in binary format (_data.dat) in rerx files.
5. Unused “Detailed Logging” option is removed from the Options dialog.

Excelerator

6. “Default [.xlsx]” now creates a more user orientated .xlsx file. Similar to “Browser [.xlsx]” except that “subtotal logic” is unwound so that each group column and it’s description appear in separate columns and that there is no Excel subtotalling.
7. Resolved bug in “query agresso template_name” functionality which was causing incorrect handling of parameters (problem was introduced in 9.4.4xx). Note: this syntax has long been deprecated and remains deprecated in favour of “agr_getbrowser template_name…”
8. Repeated misplaced continuation rows are now ignored (for compatibility with ReportEngine 8.x)
9. Resolved problem with referring to attribute columns by ID rather than by name when using “query balance …” functionality.

XtraReports

10. It is now possible to avoid embedding fonts in .pdf files output from XtraReports by specifying xf_never_embed. E.g. xf_never_embed=”Arial;Times New Roman” will cause these fonts NOT to be embedded in the .pdf even if they are used thereby making a considerable saving in file size. (ARC defaulted to never embedding these very common fonts)
11. Support for Duplex printing has been enhanced and simplified by the addition of a new “_duplex” special control. (See sample reports).
12. Resolved problem with missing “$main” table in XtraReports sometimes accompanied by “Object Reference” errors.
13. The format of “Default [.pdf]” files is improved in non-Browser/Information Browser screens.
14. The ARC to Xtra conversion tool has again been considerably improved including the ability to work with data available.

2.5. RELEASE NOTES – 9.4.602

This release contains one important new feature. Namely that it is now possible specify the “client” when logging in via the WebAPI and switch to a different allowed “client” once logged in.

There are also a number of maintenance fixes. A complete list of changes can be found below.

2.5.1. **Changes**

Common

1. It is now possible to specify the “Client” when logging in to UBW via the WebAPI connection and to select a different client during the session.
2. Resolved problem with “`Prompting parameters encountered in non-interactive mode.`” errors when using the deprecated “query agresso template_name” syntax in Excelerator reports and running the report on the report server.
3. CSV reporter now recognizes csv_final_new_line parameter to allow the final new line character to be suppressed. (See “Create CSV file.rerx” sample report)
4. A new FORMAT() function is available in ReSQL.
5. A new parameter “xf_editing_fields” is available. If set editable fields in an XtraReport will also be editable in the saved pdf output file.
6. Problem with “`Object must be of type String.`” Error when trying to display the “Export” dialog from the Template Viewer screen when the browser template contains “subtotal logic” on some, but not all, of the subtotalled columns is resolved.
7. Resolved problem with errors like “`DataTable 'PC: Subcontractor costs' does not match to any DataTable in source.`” when using agr_getbrowser against browser templates with names containing a colon “:” character.
8. Resolved problem with error “`The given path's format is not supported.`” when creating a new report in ReportStudio based on a browser template with a name containing any character that is not valid to use in a file name.
9. Resolved problem with missing file “Gramatica.1.5.dll” in the webservice installation.
10. Support for writing “.rec” data cache files is reinstated in the case where you are running a .rpx or .xlsx file directly (i.e. not within a .rerx file) on the report server queue.
11. Resolved problem with “`Failed to enable constraints. One or more rows contain values violating non-null, unique, or foreign-key constraints.`” error when reading some .rec files.
12. ReportStudio no longer opens .rec file directly. Please open a new report package and use “File>Import”. Note that opening and re-saving a .rec file in this way will prevent the above error occurring when the file is used with previous versions of ReportEngine.

Excelerator

13. Resolved problem with incorrectly trying to load sheets with Pivot table in column A in the case where said Pivot table has more than one row column.
14. Resolved problem with error like “`Multiple templates found with name 'id:169'`” when trying to run Excelerator reports with “query agresso template_name” syntax using the 9.4.601 webservice and an earlier version of the Excel Add-in.
15. Resolved problem with “Object reference” error on saving changes to Excelerator options before the “Language” is set in the “General” options tab.
16. _control sheet is now re-calculated before being read so that the current values of volatile formulas (like =Now()) are picked up and used in parameters etc.
17. Resolved problem with posting back when the postback type is left blank on the “update_columns” row (meaning accept the default of GL07).
18. Resolved problems with the treatment of “preparameter” rows in template sheets.
19. Resolved problem with “`Invalid Row Index`” error when saving .xlsx file after inserting rows above frozen region (via parameter or preparameter statements).
20. Resolved problem with the tab bar being inappropriately scrolled in output .xlsx files thus “hiding” sheets from the user until they scroll back.

XtraReports

21. none.

Worderator

22. none.

ARC

23. Problem with ARC needing to be registered on server machines is resolved. (Problem has existed since 9.4.1) (ARC does not require registration on the server machine by virtue of the fact that the COM registry settings are stored within the manifest of Agrxlcmd32.exe)

2.6. RELEASE NOTES – 9.4.601

This release coincides with UBW 7.5 and contains a number of significant key changes as well as a considerable number of smaller changes listed below. (Please note that 9.4.5xx never existed to avoid confusion with 9.4.005 which is sometimes displayed and/or referred to as 9.4.5)

Focus areas for this release are:

Support for Information Browser:
ReportEngine now makes use of the much better formatting information available from IB in UBW 7.5 in order to generate much more useful reports both when exporting from the IB screen and when using agr_getibreport within Excelerator or Resql. Even with UBW releases prior to 7.5, ReportEngine now downgrades gracefully which means it is still possible to create and run reports based in IB data.

Preparation for replacement of ARC:
ARC is based on what is now very old technology and there may come a point where it is no longer possible to support it. There is already a considerable number of issues which will grow steadily with time (please see list at the start of the "Report Creator – ARC" section of the main body of this document).
In preparation for this, "New ARC report" is no longer available in ReportStudio and the ARC to Xtra Converter has been radically improved. It is very strongly recommended that whenever it is necessary to re-visit or make any change to an ARC report it first be migrated to an XtraReport.

Completion of switch to ASPOSE implementation of Excelerartor:
The MS Excel / VSTO implementation of Excelerator is no longer available on Report Server so MS Excel will never be used on any server machine even if it is installed (which it should not be).

2.6.1. Changes

Common

1. Problem with "Unhandled Thread Exception" showing from Report Studio in the case where a "silent" login using SSO/IDS fails is resolved. (This can happen if the Access Token has expired and the user needs to log in again)
2. No longer fail to create reports based on a browser template with a non-string parameter containing a macro.
3. Selecting "Output type = Excelerator" in the UBW report ordering screen will now always create a .xlsx file containing a simply formatted dump of the data returned by the SELECT statement within the corresponding ARW report if one exists.
4. Various errors resulting from attempting to add, delete or sort rows in the data viewer within Report Studio are resolved.

5. ".SingleRowQuery" never returns more than one row. If the query returns more than one row only the first is taken, the rest are silently ignored. In previous versions, under some conditions, it would erroneously return all the rows returned by the query.
6. ".AddRow" and ".AddParentRow" are now synonymous with ".AddFirstRow". I.e. if more than one row is returned by the query only the first is taken, the rest are ignored. Previously, ".AddRow" and ".AddParentRow" sometimes, but not always, threw an exception if there was more than one row.
7. Problem with "4096 (0x1000) is an invalid culture identifier" exception when working in a customised locale/Regional settings or one of the new Microsoft supplied locales like en-NL is resolved.
8. When splitting an resql based report using the "xs_breakcol" parameter, the presence of a column named "email" or "email_to" in the query that is driving the split no longer caused the output to be emailed to that address. Note that, for backwards compatibility, this change deliberately does not affect the internal splitting mechanism within Excelerator itself.
9. The use of "WITH" prefixing "SELECT" statements is now allowed.
10. SQLite is upgraded to version 3.28. (offers support for window functions)
11. Proper support for long integers when creating SQLite databases.
12. Support for writing of .rec cache files (xc_required) is withdrawn.
13. It is now possible to specify any output extension for the CSV and XML/HTML reporters.
14. It is now possible to blank out date parameters in the parameter prompting screen.
15. Fixed "Object reference" error when prompting for parameters for a browser template with a date parameter that is left blank in the "SearchC" screen.
16. The GET_TITLES_ROW() function is introduced into dataset SQL (described in the body of this document)
17. "New ARC report" is no longer available from Report Studio.
18. Fixed issues with naming of output files for "Predefined" items on "Export" menu.
19. Fixed potential issues with invalid characters in file name when exporting from a browser template with a name containing characters such as '>"|<' etc which are not valid within a file name in Windows.
20. "Design new [.rerx]" is moved into "Design" sub-menu in "Export" menu.
21. When splitting a report and writing the output to file using the "xf_file" parameter illegal characters are now replaced with a "-" to prevent errors.
22. Formatting information is now used when "Exporting" from Information Browser Screens and when using "agr_getibreport".
23. It is now possible to load a DataSet (containing a single table) from a csv file.

Excelerator

24. Problem with incorrect outline grouping in "Export>Browser [.xlsx]" where the template had subtotal logic with columns breaking into a column which itself is subtotalled is resolved. (This is an unusual and not very meaningful construct).
25. "Export>Browser [.xlsx]" now defaults the fill colour for group footers to a different shade of grey for each level.
26. PreParameter rows no longer cause extra hidden rows in the output report sheets.
27. Problem with "invalid row index" exception when saving a .xlsx file containing both frozen panes and PreParameter statements is resolved.
28. Putting "delete" in a "hide" row will now cause the column to be deleted rather than just hidden.
29. Use of parameters in the default value given in "declare" rows now works properly. (Used to be a problem especially with date parameters.)

30. Problem with generating “default [.xlsx]” from a browser template having a name greater than 31 characters long is resolved.
31. Reporter.Excelerator.Excel.dll is no longer copied into the “ReportEngine” folder under Agresso\bin so it is no longer possible to use MS Excel to process Excelerator reports on the report server queues under any circumstance.
32. “query agr_getbrowser …” now causes the sort order to be inferred from group rows in the same way as “query agresso …”
33. Problem with password not being applied when protecting sheets when report is loaded using ASPOSE (in particular on the report server queue) is resolved.
34. Aspose implementation now follows exactly the same rules as MS Excel when it comes to recognizing strings as numbers even though this these can be a little odd at times. In particular “10,00” was previously recognized as one thousand by aspose but not regarded as a number by Excel despite the fact that Excel always has recognized “10,000” as ten thousand and “10,0000” as one hundred thousand irrespective of the number format of the current locale. This will only have any effect when loading strings into cells not formatted as text in Excel which is never recommended.
35. Problem with error “The same worksheet name already exists” when loading and distributing a report named “Sheet1” using Aspose is resolved.
36. Problem with conditional formatting rules being re-ordered when inserting rows using Aspose has been resolved.
37. Column captions are now properly preserved and used as field descriptions when generating Pivot Tables.
38. Treatment of the protection of worksheets is improved. The behaviour is now that on loading the protection of the worksheet is preserved. If a password is supplied via the “password” parameter, then this will be applied. On unload the protection is also preserved but the password will be removed. The only exception to this is that a password will never be applied to the original copy of any worksheet when loading from within the Excel Add-in. This is a deliberate feature to avoid any possibility of a password being applied that the user is unaware of.

XtraReports

39. The ARC to Xtra converter has been substantially improved.
40. Generation of new reports has been improved.
41. Report parameters are now created when generating a new report based on a table coming from a browser template.

Worderator

42. Problem with Word Mail-Merge when column names contain spaces or other non-alphanumeric characters is resolved.

2.7. RELEASE NOTES – 9.4.403

This release of Report Engine exists in order to fix issues with the WebAPI connection. A number of other maintenance issues are also addressed as listed below.

2.7.1. Changes

Common

1. WebAPI connection now requires TLS 1.1 or higher. (i.e. support for no longer secure SSL 3.0 and TLS 1.0 protocols is removed)
2. Problem with the following error *'Invalid parameter company_eq. Required parameters are: Company_eq'* when the client/company is required by a browser template is resolved.

Excelerator

3. Problem with outline/grouping opening up incorrectly after a report containing subtotals has been loaded within MS Excel is resolved.
4. Cell format is now set to “text” for string parameters in the generated “Parameters” sheet. This avoids problem with exceptions occurring if the value of the parameters looked like an Excel formula (i.e if it started with “=”)
5. Trying to insert an invalid excel formula into a non-text formatted cell is improved.
6. Generating Pivot tables based on a browser template with no amount columns no longer causes an exception.

2.8. RELEASE NOTES – 9.4.402

This release of Report Engine introduces support for Duplex printing and watermarks in XtraReports. Please see new section in the main body of this document. A number of maintenance issues are also addressed as listed below.

2.8.1. **Changes**

Common

1. Support for IDS/Single Sign On is improved.
2. Problem with creating new ODBC connect string from within Login window is resolved.
3. Problem with “Email exceptions” report not being loaded into asysprintblob (and no distribution report being created) in the case where the report is based on an rerx is resolved.
4. Various problems with importing reports against Browser Templates and Information Browser Reports are resolved.
5. The number of stored urls visible in the login window is increased from 25 to 255.
6. Problem with “.splitquery” only being recognized if the table name was “main” is resolved. (The whole idea is, of course, that “.splitquery” allows you to give a meaningful name the table which will drive the splitting of the query)
7. The “$$dbtype” system parameter now behaves as it should within ReSQL
8. Problem with 8-digit strings like ‘30191201’ being incorrectly interpreted as dates in RE SELECT statements has been resolved.

Excelerator

9. Subtotal logic problems with “Browser [.xlsx]” within UBW desktop where the “break into” column has been renamed or the subtotal column is set to “Code/Text” are resolved.
10. Problem with “*Row index and column index cannot all be zero*” error when processing .xlsx file in which the frozen row or column is to be deleted. Instead the sheet is now unfrozen.
11. “Load Current Sheet” now works properly when the sheet name contains commas or quotes or both.
12. Problem with “Invalid Pivot Field Name” errors after loading a report containing ordinary Excelerator report sheets which are being used as the data source for a Pivot table is resolved.
13. The “tab” at the bottom of the screen in Excel for the active sheet is now made visible at the point where the sheet is activated.
14. Problem with Excel crashing when opening .xlsx file containing Pivot Slicers on numeric non-integer data which has been loaded under a locale (number format settings) different to that under which it was created is resolved.

XtraReports

15. Generated reports now have padding applied to label controls in the same way that they would if they were created in the designer itself.
16. Support for duplex printing and watermarks is added. (somewhat similar to canvas [.cnv] files in ARC). See “Duplex Printing and Watermarks” section of this document.

17. Various improvements to report validation process. In particular, expressions containing aggregate functions no longer generate validation errors.
18. Report validation is now only applied within the report designer. I.e. it is no longer applied at run-time and consequently validation warnings will no longer be printed in the log file.
19. New generated reports never contain “Detail Reports”. Instead the relational structure of the data is made available by using the “$tablename” feature resulting in a much simpler and more correct report structure.
20. Changes made to the layout of the panels within the XtraReports Designer are now persisted in the registry so that the designer will have the same layout the next time it is opened.
21. A number of new sample reports are included.
22. Standardized names for group headers and footers now include the group level which makes it easy to see matching group headers and footers.

2.9. RELEASE NOTES – 9.4.302

This is a service update of ReportEngine release 9.4.3 that we have made to resolve small number of important problems, principally changes 1,2,3,13 and 14 below which have been reported or discovered in 9.4.301. A relatively small number of other changes are also included as listed below.

2.9.1. **Changes**

Common

23. A scroll bar has been added to the parameter prompting window to cater for cases where there are a very large number of parameters.
24. Problem with sizing of Login and other dialogs when using very high resolution monitors is resolved.
25. Bug in agr_gettext (and agr.GetText() from ARC) is fixed. This bug was introduced in 9.4.301 and caused agr_gettext to return blank or in exceptional cases the wrong text.
26. In the "Data Schema Viewer" in Report Studio the hierarchy of the tables now shows the deepest child tables at the first level (left) in the structure in sympathy with the recommendation to use the new "$tablename" feature.
27. When using "Run Package" from Report Studio "data0" is now read properly from _data0.dat if such a file exists.
28. It is now possible to save _data.dat to .sqlite or .xlsx from within Report Studio. It is also possible to import _data.sqlite or _data.xlsx and these will be converted directly to _data.dat.
29. Problem with ".query main RE SELECT * FROM data0.*tablename*" resulting in table *tablename* being removed from the data0 dataset and consequently unavailable to subsequent queries is resolved. Note: this problem was introduced in 9.4.301 and only affects this exact and unusual case, i.e. a trivial SELECT of all columns and all rows from a specific table into query "main".
30. It is now possible to specify a full ODBC connect string on the agrxlcmd command line using the new "/odbc:connect_string" switch.
31. Full provider strings and odbc connect strings are no longer shown in the log file as they might contain passwords and other security sensitive information. Previously the only time they were shown was if they were provided on the agrxlcmd command line.
32. It is now possible to log in "silently" when connecting via IDS.
33. There is a new "xs_logall" parameter which allows the user to force logging to be included for every iteration of a split report. If missing or set to 0 then only the first, last and every 1000th iteration is logged. (xs_nolog no longer has any affect)
34. Problem with texts in "Reports" menu within UBW Desktop always showing in English has been resolved.

Excelerator

35. Problems with "Frozen Panes" being incorrect after loading an Excelerator report are resolved.
36. "Default [.xlsx]" has been re-instated on the Report Engine menu available from UBW Browser/Template Viewer and Information Browser windows.

XtraReports

37. The XtraReports samples are improved and updated. They all now use the new "$tablename" technique which is to be recommended over a nested "DetailReport" structure in almost every case.
38. A new file called "ARC to Xtra Migration Guide.pdf" is included in the installation.

39. When using “$*tablename*” to denormalize data for a report, parent columns in a relation are no longer omitted and defensive code is added to allow for the possibility of duplicate column names.

2.10. RELEASE NOTES – 9.4.301

This is an important release of Report Engine for a number of reasons:

- Report Studio stability issues are addressed.
- The ASPOSE implementation of Excelerator is completed
- The XtraReports Designer and Runtime have been radically improved.
- The “Silent Login” from the Excelerator Add-in has been re-instated.
- Support for logging in using IDS (3.0) is added.

2.10.1. Changes

Common

40. A number of serious stability issues that previously caused crashes and hangs in Report Studio are resolved.
41. A new parameter named xs_allow_changes is available. If defined and set to “1” then it is possible to change the value of the xe_* and xd_* parameters AFTER .query main / .splitQuery xxx in the resql. E.g. it is no longer necessary to include the email address within the main query which under some circumstances enaables a huge performance increase.
42. It is now possible to use IDS with the WebAPI connection to Unit4 Business World.
43. Problem with errors like “hexadecimal value 0x1A is an invalid character” when trying to read data containing such characters over the web service connection is resolved.
44. Problem with being unable to remove temporary folder after opening the help file from within ReportStudio is resolved.
45. Problem with “no data available” message appearing incorrectly within some UBW desktop screens is resolved.
46. A number of problems resulting in an error like… “prompting parameter ‘????’ encountered in non-interactive mode” are resolved.
47. There is a new item on the “tools” menu in Report Studio to remove all data from _data.dat leaving only the data schema. This allows XtraReports (.repx) to be edited/designed without having to have any potentially sensitive customer data in the .rerx file.
48. Problem with “argument cannot be null” exception in “RE Select … GROUP BY xxx” where there are no rows selected is resolved.
49. Problem with the value of $breakcol parameter being incorrectly given when splitting on more than one column is resolved.
50. A number of issues connected with the user being taken back to the wrong window after closing a dialog box shown by Report Engine are resolved.
51. Save As… is now available for individual files within the package explorer.
52. Problem with unclear (fuzzy) text in Report Studio when using a high resolution display is resolved.
53. Web service has been updated so as not to cause problems with “Kerberos”
54. New resql keyword “.splitQuery” to the user to specify a name other than “main” to the table that will drive the splitting process.
55. New resql keywords “.addParentRow” and “.addChildRows” which are simply synonyms for “.addRow” and “.addRows” respectively.

Excelerator

56. By default, all reports are processed using ASPOSE. The only way to revert to using VSTO is by explicitly setting the x_xlreporter parameter to VSTO on a report by report basis – this is not recommended.
57. User is no longer forced to see the “Log in” screen when using Single Sign On. Instead the “Silent Login” functionality from previous releases is re-instated.
58. Implementation of Pivot Tables is overhauled so as to be much more sympathetic with the rest of Excelerator.
59. It is now possible to create a Pivot Table based on a browser template directly from within Excel.
60. Exception caused by supplying a password to unprotect a sheet where none is needed using Aspose is resolved.
61. Problem with writing into Formula property of cell formatted as text when using Aspose is resolved. Now behaves exactly as the old VSTO implementation did.
62. Problem with deleting control information (column A) from sheet where data has been loaded into an Excel “Table/ListObject” when using Aspose is resolved.
63. “Browser [.xlsx]” function is now implemented using Aspose rather than SpreadsheetML. It now runs up to twice as fast and produces a file that will be displayed properly in “LibreOffice” and other non-Excel .xlsx viewers.
64. Problem with ReportEngine incorrectly believing the MS Excel is installed (and trying to use it) on a machine on which it was once installed but has now been removed is resolved.
65. Problem with case sensitivity and use of display name in “sheet” parameter is resolved.
66. When creating pdf output from Excelerator the pdf file can now be password protected and encrypted. (NB this only applies when Aspose is being used)
67. Problem with “header, page” putting the page-break after the current row rather than before it is resolved.
68. Output sheets are now presented scrolled to the upper-leftmost visible cell in a consistent manner. Previously sheets were often scrolled away from with point with apparently random cells selected.

XtraReports

69. If the DataMember for a report is set to “*$tablename*” rather than “*tablename*” main then the Xtra Reporter will expand the named table to include columns from all parent, grandparent etc tables. All parent tables so included are also removed from the DataSet as seen by the Report Designer. This has the effect of making it hugely simpler to select the correct data fields.
70. The “parameters” table is no longer visible in the XtraReports Designer “Field List”. Instead all parameters existing in the DataSet (.declare’d in the ReSQL) are copied into the XtraReports “Parameters” when the report is generated and also be the new “Auto Correct Parameters and Data Bindings” tool. This removes confusion and makes sure that all parameters are available for use in Calculated Fields and Conditional Formatting Rules which was not previously the case.
71. The ARC to Xtra converter has been greatly improved. In particular, it makes use of the above “*$tablename*” feature to generate an XtraReport that has exactly the same structure as the original ARC report and which can run from exactly the same DataSet. It will no longer generate the nested “DetailReport” structure which was very difficult to work with and not always capable of creating the required report output.
72. There is a new “Rename Data Members / Tables” tool which allows for fixing up the report after a table is renamed in the ReSQL or a report is converted from the “Nested Detail Report” model to the “*$tablename*” model.
73. The Report Validation process has been very much improved. Unfortunately, to the extent that some (many?) existing reports now give validation errors that were not previously picked up. To mitigate this, not all validation errors are now fatal at runtime. But please be aware that ALL validation errors and warnings are important and should be resolved.

74. Data is now refreshed automatically when needed within the XtraReports designer.
75. Output file type “.docx” is now supported.
76. Password is now applied (if given) when creating “.xlsx” or “.xls” output files.

Worderator / MailMerge

77. Use of MS Word Mail-Merge is much simplified and separated from the rest of the Worderator Add-in
78. The Worderator Add-in is no longer “Package Aware” in the same sense as the Excelerator Addin. I.e. it no longer knows anything about .rerx or .resql files.

XML Reporter

79. The processing of XSLT has been improved to support xsl-output configuration

2.11. RELEASE NOTES – 9.4.202

This release contains fixes to some important maintenance issues as well as substantial improvements to the XtraReports integration. Anyone developing XtaReports with any previous version of ReportEngine 9.4.x should definitely take this release otherwise their reports will not run on a target machine with any lower version of ReportEngine installed.

2.11.1. Changes

Common

1. Problem with invalid cast exceptions when writing cache files (xc_required=1) when running on server queue.
2. In RESQL parameter substitution, single quotes are now only escaped in contexts where the whole parameter is enclosed in single quotes. I.e. '$?obrien' where the value of parameter obrien is O'Brien will be escaped whilst $?sql_fragment will not.
3. Font scaling within Report Studio is improved.
4. Problem with "No Data Available" message appearing incorrectly when pressing "Export" from TopGen screens is resolved.
5. Problem with "LUHN()" function when given strings with more than about 20 characters is resolved. Also a new scalar function called AppendLuhn() (case insensitive) is added which returns the input string with the check-digit appended.
6. On opening .rerx files ReportStudio now detects orphaned temporary working folders and performs an Auto-Recover if any are found. Recovered files are placed in "My Documents" with names like "Recovered_???.rerx"

Excelerator

7. Problem with the use of formulas which evaluate to "$sum" in subtotal footers is resolved. This now behaves as it used to in Excelerator 8.x.
8. Creation of .xlsx files containing Pivot tables and data sheets is greatly improved.
9. The Excelerator Excel Add-in no longer has any knowledge of Report Packages (.rerx files) and .rerx files containing .xlsx files will open with ReportStudio so that it is possible to work with them in exactly the same way as .repx (XtraReport) files. (It is possible to revert to the previous behaviour by creating a string registry value called PACKAGE_AWARE_ADDIN under HKEY_CURRENT_USER\Software\UNIT4\ReportEngine\Excelerator with a value of "1").
10. Problem with screen refresh during display of "Login" and "Options" dialog on Windows 7 is resolved.
11. Problem with not being able to Login or display Options whilst in "Edit Mode" is resolved.
12. Problem with writing data sheets into .xlsx file when any column or parameter name contains a space is resolved.
13. ASPOSE is now used by default for processing of Excelerator reports being run within the UBW desktop via the "Reports" menu. (For symmetry with the same report potentially being run from the UBW web application where this is the only possibility)
14. Problem with opening a report package (.rerx file) containing an Excel workbook with the same name as a workbook that is already open is resolved. (previously the package was not opened and nothing happened)
15. Timeout is now applied properly when using the deprecated "query AGRESSO *template_name*" syntax. (previously the timeout was only applied when using the current "query agr_getbrowser *template_name*…" syntax.

XtraReports

16. XtraReports / .repx files are now serialized as xml rather than as code to avoid inter-version compatibility issues. (Without to this change XtraReports created with 9.4.1xx could not be opened with previous versions of ReportEngine)
17. Problem causing Tools>Refresh Data in XtraReport designer to refresh only the data in the field list but not the rest of the report is resolved.
18. Problem causing sub-reports not to contain data when previewed in the designer is resolved.
19. Data is now properly attached to reports opened directly from within the XtraReports designer rather than from Report Studio.
20. The ARC to Xtra converter has been improved.
21. Validation errors are no longer incorrectly reported when using “Mail Merge” style fields with format specifiers or if top level table name contains a dot “.”
22. “default [.pdf]” report no longer fails if the browser template name contains a dot “.”

2.12. RELEASE NOTES – 9.4 SERVICE UPDATE 1A

Service update 1a exists only to correct a small number of important problems discovered in the installation image of service update 1 which could not be corrected without a new update.

2.12.1. Changes

Common

1. Several DevExpress dlls were missing from the installation package which gave rise to errors when using the XtraReport designer.
2. Problem introduced in service update 1 with “query balance” excelerator reports using syntax like “WHERE account 123;456” is corrected. The presence of the semi-colon (meaning OR) was causing errors like “column ACCOUNT not found”.
3. Problem with incorrect behaviour when exporting from browser templates in the web application (Template Viewer) where the template has subtotals but no numeric/summable columns is resolved.
4. The documentation concerning xs_password, xw_password and xf_password is corrected.

2.13. RELEASE NOTES – 9.4 SERVICE UPDATE 1

2.13.1. Headline features

This service update addresses a considerable number of maintenance issues as listed below. It also contains one important new feature that is DISABLED BY DEFAULT so as not to present an obstacle to upgrading for maintenance reasons. All types of Excelerator report can now be run on server machines without having to install MS Excel as discussed below in the section entitled “ASPOSE CELLS”.

2.13.2. Changes

Common

1. Problem with non-numeric address types when using agr_getaddress is resolved.
2. When importing data in xml format (_data.xml) into ReportStudio it is now correctly transformed to binary format (_data.dat).
3. ReSQL .set and .declare statements have been enhanced to allow expressions in the given value.
4. When using “Default [.xlsx]” option from Browser/Template Viewer/Information Browser empty tables no longer trigger a problem with invalid Pivot Tables.
5. In the event of a failure to add a document to the archive the distribution status is now set to “failed”. (such failures used to be ignored)
6. Added new command line diagnostic tool to show registry entries governing Excel addins (“agrxlcmd registry /log:logfile.txt”)
7. AgrxlInstall log files are now placed in the TEMP\ReportEngine folder like all other log files
8. “AgrxlInstall register” now has a new /64 command line switch to register in the windows 64 bit registry

9. Problem with display of Report Engine “Export” menu in Information Browser pages when using Internet Explorer 10/11 is resolved.
10. Problem with creating immediate parent folder when using output_dir/output_file is resolved.
11. ODBC passwords are now properly obscured again (problem introduced in 9.4)
12. Problems with “Login on demand” are resolved.
13. Two new parameters named agresso_export and agresso_import containing the values of the corresponding environment variables are available when running on the report server.
14. Log file now only ever shows the full details of the first iteration when splitting reports resulting in a much smaller file sizes.
15. Parameters containing “<parameter_name>” syntax are now all substituted properly before being passed to the Report rendering process.
16. New parameters xs_password and xw_password are introduced to allow separate control of the password applied to the whole report output and the individual split report outputs.

Excelerator

17. Several problems which resulted in the “Unit4 Excelerator” ribbon not appearing in Excel have been resolved.
18. Resolved problem with incorrect “Nested Subtotals” exception when posting back if one subtotal row immediately follows the footer of another subtotal row.
19. Improved exception message when multiple detail or group rows encountered within subtotal structure
20. Problems with missing Agresso.ReportEngine.AgrServer.manifest file are now properly logged to the log file.
21. If user selects “Output Type=Excelerator” for a server report where no .xlsx file exists in “Customized Reports” the report is now run in-process and will not be subject to a timeout of 30s.
22. Output Type = “Excelerator” now works on 64bit report server queue.
23. Output Type = “Excelerator” now creates a “.xlsx” file rather than a “.xls” file. (Important if there are more than 65k rows)
24. SetPrompt now correctly handles prompting when posting back (problem introduced in 9.4)
25. Problems with parameter / preparameter rows where some cells on the row are merged are resolved.
26. Problem with reading language ID from Excel is resolved. (Previously caused failure to initialize Excel process without meaningful exception message)
27. There is a more consistent approach to which sheet will be the active sheet after loading.
28. Excel outline/grouping is no longer generated if source browser template does not have summary rows.
29. It is now possible to Process fully featured Excelerator reports without involving MS Excel. (See discussion of Aspose Cells below)

ARC

30. Problem with viewing “rdf” files due to “arpro2.dll” being missing is resolved.

XTRA

31. ReportEngine now uses XtraReports version 17.2
32. ARC to Xtra Converter tool has been improved.
33. An unbound single-row report containing DetailReport bands with non-blank data member no longer causes a validation exception.

2.13.3. ASPOSE CELLS

The ability to run Excelerator reports on the UBW server queues (ordered reports) is a very powerful and widely used feature. However, with time the use of MS Excel on server machines in this way is becoming very difficult to support from technical, installation and licensing perspectives. For this reason, we have invested considerable effort in creating an alternate solution using a third party API known as ASPOSE CELLS which manipulates .xlsx files directly without involving MS Excel. All existing reports will run and yield exactly the same results as they always have. Users should notice no difference except that in most cases their reports will run substantially quicker.

There are no licensing or installation issues associated with ASPOSE. All necessary dlls are shipped with the ReportEngine installation and no run-time license is required.

However, this is a substantial change and this is a service update and so it is DISABLED BY DEFAULT in order not to present an obstacle to users wishing to take advantage of the considerable number of maintenance issues fixed in this release. If this update is installed on an existing machine the ASPOSE integration will not be used at all except:

- to process a “full functionality” Excelerator report imported against a browser template or information browser when running within the UBW web application. (These scenarios would have failed in previous releases).
- If you have an Office 365 / Click to Run installation of Excel on your server machine. (This should only be the case for test/demo installations of UBW where the server is actually installed on a client machine)

The new functionality can be enabled in 3 ways…

1. By not installing MS Excel on the server machine (or having only a “Click to Run” installation of Excel downloaded via an Office 365 subscription).
2. By setting a system parameter RE_XL_REPORTER to a value of “aspose”.
3. On a report by report basis by setting a parameter called x_xlreporter=”aspose” in the report variant or in resql.

For all new installations we would urge you to take the first “do nothing” approach. I.e. do not install MS Excel on any new server machine. On existing installations, we suggest using the third possibility to test individual reports and take advantage of performance increases.

NOTE 1:
It is possible to force the use of a “Click to Run” installation of Excel downloaded via an Office 365 subscription by setting the system parameter RE_XL_REPORTER to a value of “vsto”. BUT you should only ever do this for test purposes (it will only work if running as “Local System” and there are various other limitations).

NOTE 2:
The ASPOSE integration is never used from within the Excel Add-in. There is no change to the processing of Excel reports when opened interactively in Excel.

NOTE 3:
In future releases this is likely to become the default, and possibly the only, way of processing Excelerator reports on server machines.

2.14. RELEASE NOTES – 9.4

Report Engine 9.4 includes a number of new features as well as fixes to important maintenance issues.

2.14.1. Headline features

- Support for Unit4 Business World On! Spring 2017 (aka Milestone 7)
- Substantial improvements to "Predefined" (automatically generated) report output from Desktop Browser, Template Viewer and Information Browser including…
  - Completely new "Pivot [.xlsx]" option that creates an Excel Pivot table of the current data
  - The addition of a "Parameters" sheet to the "Browser [.xlsx]" option
  - Complete overhaul of the "Default [.pdf]" option
- Standalone Excelerator reports (.xlsx not .rerx) now work on 64bit queue. This is important both because it means that there is no need to change such reports and also because it preserves support for the Post-Back functionality.
- All reporter types (except ARC on a 64bit queue) now run in-process on the report server which represents a substantial performance improvement when splitting report output. (this facility was introduced for XtraReports only in 9.3 su2)
- The Excelerator Ribbon is greatly simplified when dealing with standalone .xlsx file (as opposed to a report package / rerx file).

2.14.2. Changes

Common

1. Support for Unit4 Business World On! Spring 2017 / MS7. Including integration with Information Browser and deployment within Microsoft Azure.
2. "client" parameter is now correctly passed to report when "run from menu" is set in UBW Smart Client.
3. Exception is no longer thrown when invoking ReportEngine from a Smart Client Browser screen with a browser template containing a prompting parameter named "client" or "period". (This problem was introduced in 9.3su2)
4. Problem with splitting report data where break column contains nulls is resolved.
5. On UBW report server setting "xr_required=1" now causes the output .rerx file to be inserted into the database so as to be available in "maintenance of ordered reports"
6. The database timeout is now respected when using agr_getbrowser
7. Agr_getaddress now returns an extra column called formatted_address3 containing the address in single line format suitable for printing at the bottom of a page in a report.
8. No timeout is applied to the report rendering process (Agrxlcmd.exe) of a rerx based report when running on the business server. This is so as to be consistent with the fact that no timeout is (or has ever been) applied to a standalone report on the business server and avoid problems with reports timing out after being migrated.

9. Rendering of all report types is now run in-process on the report server (except ARC reports running on a 64bit queue which is not possible). This removes the overhead of creating a new process and can have a very large effect on the execution speed when splitting reports.
10. It is now possible to control the Locale / Culture settings used for any report by defining and setting a parameter named “locale” in the resql.
11. Temporary split output files created as a result of setting xf_file but not setting xf_dir are now properly removed.
12. Any report with a (report rendering) timeout > 30 seconds will be streamed to the user as a report package (.rerx file) to be rendered locally. This is the same behaviour as from TemplateViewer screens if the ReportEngineLongAsRerx application setting is active.
13. There is a new type of web service connection to Business world following the “WebAPI” technology.
14. Consistent naming of unsaved report packages in Report Studio.
15. There is no longer an option to view the log file when running from the web application unless a system parameter named RE_LOGGING is created and its value set to “ALL” or “DESIGNER” (to allow access for designer users). So by default the log is not available. This is to avoid “Technial Information Leakage” which could under some circumstances be viewed as a security issue.
16. The treatment of Boolean values returned from Desktop Browser screens is corrected and now agrees with the data returned from the same template viewed on the web via TemplateViewer. (Previously the value was inverted)

XtraReports

17. Problem with Report Setup process not finishing properly certain validation errors were encountered is resolved. Symptom was that the “Field List” in the report designer was empty.
18. SectionName is now shown as well as ControlName in validation errors.
19. It is now possible to set the permissions on pdf output files (see “special parameters”)
20. Version of XtraReports is upgraded to version 16.2 which fixes a number of issues.
21. New “Review Control Names” tool available in XtraReports Designer
22. It is now possible to re-read the data within the XtraReports Designer so that it is no longer necessary to close and re-open the designer to see changes to the data.

Excelerator

23. Problem with outline/grouping not being shown properly in output sheets when using the “break” keyword or “group” keyword in the “_control” sheet is resolved. (this problem has been there in all previous 9.x vesions)
24. Problem with no error being raised when attempting to use wildcard comparators in “WHERE” rows against a numeric data column is resolved.
25. All COM Add-ins are now disabled temporarily whilst invoking Excel as an out-of-process server (principally on the report server)
26. In the event of a failure to create an instance of Excel much more diagnostic information is written into the log file.
27. ForceFullCalculation is no longer set when generating .xlsx files using OpenXML (e.g. “Browser [.xlsx]”) as this caused the instance of Excel used to open them to go into FullCalculation mode and remain in that mode until the whole of Excel was closed and re-opened. This meant that every change (e.g. typing a single character) caused a full recalculation of all open workbooks.
28. Problem with “invalid token ‘order by’” in “query balance” reports without “relation”, “tree” or “where” keywords is resolved.

29. Standalone (non-rerx) Excelerator reports (.xlsx and .xlsm) files can now be run on a 64bit report queue if they are found in “Customised Reports”.
30. Postback to PR28 / aprimport and LG20 / algproductinput is now enabled.
31. Postback no longer fails if table that we are posting to contains columns of type “GUID”
32. ParserLevel now defaults to “DataSet” for all .xlsx files contained within .rerx files.
33. Behaviour of “Options” window is improved. (Previously it was possible to be editing the options for a workbook other than the currently active one)
34. There is a new “Keep Control Information” option available which causes the control information (Column A plus control rows) to be kept even at run time.

Worderator

35. The Worderator “Mail Merge” integration functionality has been substantially re-written. In particular the problems with warning messages from MS Word saying “Opening this document will run the following SQL command” followed by an error if you answer “yes” is resolved. Note that existing rerx files may still exhibit this behaviour until they are re-saved.
36. Headings in header row in default generated tables are now taken from the column caption rather than directly from the column name (which was often meaningless)

ARC

37. Problem with “method not implemented” exception when generating new rpx file from ReportStudio is resolved.
38. ARC reports will now respect the “locale” parameter if defined in resql / report parameters. But note that in the case of ARC the values should be set to the numeric locale id like “2057” rather than the locale name like “en-GB”. (Other reporter accept either)

CSV

39. There is a new parameter available named “csv_nobom” which can be set to supress the “byte order mark” at the beginning of the output .csv file.

2.15. RELEASE NOTES – 9.3 SERVICE UPDATE 2

9.3 Service Update 2 contains further important enhancements to the XtraReports functionality, support for Unit4 Business Wolrd MS6su1 as well as fixes to a number of important maintenance issues.

Of particular note are changes 2 and 9 below.

Running XtraReports “In Process” on the UBW Business Server results in a very considerable (often tenfold or greater) reduction in the overall execution time when server reports are being split and distributed. Note that this applies to XtraReports, XML and CSV reports only and not to ARC.

ARC to XtraReports converter. Within Report Studio it is now possible to right-click on a “.rpx” (ARC) report and select “Create repx” which will automatically generate a “.repx” (XtraReports) report. The converted report will include all controls/fields in the correct positions in the report. As far as possible the structure of the report will be correct. However, this does depend on the corresponding DataSet being in a good shape to start with. VBScript from the ARC report is not migrated and must be replaced by “Calculated Fields” and/or “Conditional Formatting Rules”.

2.15.1. Changes

Common

1. Support for Unit4 Business World MS6su1
2. XtraReports, XML and CSV reports are now rendered “In Process” when running on the UBW Business Server thus avoiding the very considerable overhead of creating a new process for each iteration of the report splitting process.
3. No attempt is made to “InsertReportFile” when client type is not “Business Server”
4. Problem with data extracted from .arw not being included in .rec file is resolved.
5. Problem with removing temporary tables with relations during the execution of RESql is resolved.
6. Open file location: under Run Report within Report Studio there are now two options: "Open file" and "Open file location", which allows the user to either open the output file directly or display the folder in which it has been generated.
7. SQLite provider has been enhanced including proper treatment of “$” functions in SQL statements and blobs/images within the SQLite database. Column names returned from SELECT statements no longer include the table prefix. I.e. SELECT t.client FROM acrlient t … will now result in a column called “client” as it would on any other database rather than “t.client” as it did previously.
8. GROUP BY has been added to the RE SELECT … statement.

XtraReports

9. ARC to Xtra converter. It is now possible to generate a new XtraReport (.repx) file from an existing ARC (.rpx) file. The report layout and structure is properly converted but VBScript is not converted.
10. There is now much more validation of reports before they are run. In particular of the data bindings.
11. Support for the “pdfa” / “archive” format has been added. To obtain a pdfa compatible pdf file the new xf_pdfa parameter must be set to “1”.
12. Pdf file properties (Subject/Author/Title) can now be set via the xf_subject, xf_author, xf_title parameters.
13. A greater number of sample reports are included in the release

Excelerator

14. Worksheets are now always re-calculated immediately after they are loaded. This resolves problems when referring to cells containing formulas in one loaded sheet in another sheet that is being subsequently loaded.
15. Problem when using Boolean data in a “Code” column in Excelerator is resolved.
16. Control characters are now removed from data before inserting into xlsx files to prevent errors like: “hexadecimal value 0x15, is an invalid character.”
17. It is now possible to post to a client other than the logged in client when posting back to CS15/acsheaderinput

Worderator

18. Problem with Worderator generating an invalid .docx file when zero rows are returned into a table in the document is resolved.

2.16. RELEASE NOTES – 9.3 SERVICE UPDATE 1

9.3 Service Update 1 contains a very major update of the XtraReports functionality but is a minor service update of the rest of the product. Anyone who intends to develop or deploy XtraReports should be using this release.

2.16.1. Changes to XtraReports

1. XtraReports is upgraded from 13.1 to 15.2. Note: an implication of this is that any XtraReport (.repx file) created with this version of Report Engine will not be readable with any previous version of Report Engine. The improvements in XtraReports itself between these versions are certainly great enough to justify this inconvenience. (A .repx file created with previous versions will open in this version without any problem)
2. Added validation of data-bindings at the point where the report is opened. At design time this lists any errors in the log file but allows the user to continue. At run time any such validation error will cause an exception and execution will stop. Note: this implies that any such incorrectly written reports created with earlier versions of Report Engine will now fail immediately. This is deliberate and intentional functionality – it is much better to fail immediately than allow a report to run which will give incorrect results.
3. Added support for subreports in XtraReports
4. Improved user experience when working with the XtraReports Designer
5. Generated Reports are greatly improved and allow the use to choose which table the report should be based on. (Useful for creating reports that will be used as subreports as well as the main report)
6. Default/Predefined pdf output files are now created with XtraReports rather than with ARC as in previous versions.
7. XtraReports Samples and Documentation have been updated.

2.16.2. ReSQL Changes in Support of XtraReports

1. Added ".singleRowQuery" in ReSQL which allows use of the "single row join" syntax outside of ".addRow"
2. Added "CONCAT()" aggregate function in ReSQL which concatenates string values from individual rows into a single field. (e.g. for creating invoice notes from "short_info" columns in Business World)
3. The agr_getaddress macro now creates an additional column named "formatted_address2" which is the same as "formatted_address" except that blank lines are removed.
4. The names of relations generated by ReSQL are changed from *parentTable_childTable* to *childTable* or *childTable2* etc if there is a conflict. This makes the data explorer and data member names in XtraReports much shorter and easier to read.

2.16.3. Other Changes

1. Fixed “Run Package” in Excel and Word add-ins
2. Performance when reading and writing SQLite database improved several hundredfold.
3. Fixed “This command is not available because no document is open” problem that sometimes occurred from the Worderator Add-in when opening a Word document from outlook is resolved.
4. User is no longer logged out when loading an Excelerator sheet with parser level set to “DataSet”
5. Slight change to the behavior of xc_required. In the case of a .rerx based report an output .rerx file including the current data will be created (usually in “Report Results”) rather than a .rec file.

2.17. RELEASE NOTES - 9.3

This release contains some important changes including the fact that it is the first release with support for Unit4 Business World Milestone 6 and that there has been some “re-branding” work so you will see changes to the appearance most of the screens. However, it also contains a number of other important changes.

2.17.1. Major Changes / Highlights

1. Added Support for Unit4 Business World Milestone 6.
2. Report Studio has been considerably enhanced and now has a UI much more similar to the Excelerator and Worderator add-ins.
3. It is now possible to run the complete report package (.rerx file) from within Report Studio and the Excelerator/Worderator add-ins in the same way that the report would be run on the Business World report server including the report splitting.
4. Some performance issues and important bug fixes in Excelerator are resolved (listed separately below).

2.17.2. Excelerator Changes

5. Support for Office (Excel and Word) 2007 is removed.
6. Installation problem with “SetDCOMAppId: Object reference not set to an instance of an object.” is resolved. This occurred if the Excelerator AppId was not present in both the 32bit and 64bit registries. After this failure Excelerator would not run properly on the Report Server.
7. Problem with cells containing values formatted as dates not being recognized as dates in criteria strings (code columns and crosstab rows) is resolved.
8. Problem with “Call rejected by Callee” errors when opening a workbook requiring a lot of calculation (often because it contains Pivot tables) is resolved.
9. Fixed performance problem when using wildcard comparators in Excelerator.
10. Improved performance especially when loading subtotal structures.
11. When reading data from data sheets in Excel empty cells now get loaded as blank/zero length string rather than as null.
12. Preparameter rows are now properly unloaded in template sheets.

2.17.3. Other Changes

13. “Percentage” columns (or other columns where Vertical Sum = False) in Browser Templates no longer have sums/subtotals in generated “Browser [.xlsx]” or “Predefined [.pdf]” files.
14. Problem with extra DataRelations being created when a .loop structure had more than 3 nested groups is resolved.
15. Excelerator and Xtra reports now respect xf_password parameter when splitting reports.
16. Fixed issue with substituting parameters in resql where the first character after the parameter is a double quote (“) or a closing square bracket (]).
17. Error handling when processing resql has been improved and now shows the line number in a much larger fraction of cases.
18. Introduced support for running reports against SQLite database files and also for saving data to SQLite database files.
19. The “new Xtra Report” functionality has been improved to generate a report supporting images and with better named sections.
20. It is now possible to declare prompting parameters in ReSQL without providing a default value for the parameter.

2.18. RELEASE NOTES - 9.2

This is a major new release of Report Engine that introduces 2 new report rendering components completely new functionality in Worderator and “Xtra Reports”. Excelerator is also considerably enhanced by a new UI making use of the office ribbon and custom panels within Excel.

Major Changes:

1. The Excelerator Add-in has been substantially updated with a completely new UI and support for the new Report package file model.
2. Substantial new functionality is available in Worderator.
3. The Xtra Reports report renderer is now included.
4. The ReSql editor is very much enhanced and made available as panels within both the Excelerator and Worderator Add-ins.
5. Excelerator performance has been substantially improved especially when using Excel 2010 or 2013.
6. Support for updating data sheets within existing .xlsx files without using MS Excel is added. This allows powerful reports (e.g. including Pivot tables) to be processed on the web-server and delivered to the user without requiring any client based software.

Compatibility issues:

1. We no longer support Agresso 5.5.3 and Unit4 Agresso Milestone 1 and 2.
2. Support for “direct connect” ARC reports (i.e. those containing SQL calling back to the database) running on Agresso Web (classic) is withdrawn. Such reports must be migrated to get their data via a resql file.
3. Support for the historic and undocumented Excelerator “File List” functionality is removed.
4. Support for Excelerator “Insert Strings” (Column E in sheet row in _control sheet) is removed.
5. The “Parameters as Strings” option is removed from Excelerator. Parameters are now always given their properly typed values. (This option has defaulted to false for all new reports since Report Engine 8.x)
6. Support for the <t_*titlename*> syntax in Excelerator is removed.
7. The “Extract” keyword is removed from Excelerator.

Known issues:

8. In order to use old Worderator/MailMerge reports it is necessary to open and re-save them within Report package [.rerx] files.

2.19. CHANGES IN SERVICE UPDATE 1

1. A new Add-in called ExceleratorVBAInterface.xlam is provided to allow user-written macros to communicate with Excelerator in the same way which was previously possible. The macro names in this add-in are the same as in the old agrxl8.xla add-in so the only change that is required in an xlsm file which uses these macro entry points will be to change the reference from agrxl8.xla to ExceleratorVBAInterface.xlam.

2. Support for “=AgrGetDescription()” worksheet function is re-instated via the new ExceleratorVBAInterface.xlam Addin.
3. Problem with agrxlcmd child process not finding input files is resolved. This either resulted in a simple “input file not found” exception or to a timeout when running from Agresso Web. The log file from this process is now flushed to disk as it is written so as to be visible in the event of the process timing out.
4. Problem with “Login” parameters (“client”, “user_id” and “language”) being missing from Excelerator reports is resolved. (SDE 20067).
5. A new “Data Explorer” window is available from the Excelerator add-in, Worderator add-in and from Report Studio. This window is aware of relations defined in the dataset and allows the user to view the data in a hierarchical way. This is especially useful for Worderator and XtraReports which both make heavy use of the same relations in the dataset.
6. Several problems with the installer package have been addressed.
7. Problem with tables defined by datasheets not being available after Load>Select… in Excelerator is resolved.
8. User is no longer forced to login when loading an Excelerator report if the Parser Level is set to “DataSet only”.
9. When creating .xlsx files using OpenXML Excel “Named Ranges” are now created (or updated) to point to the entire range of data. This is very useful when basing Charts of Pivot tables on this data.
10. “OpenXML” type .xlsx reports are now opened properly when run from Agresso Desktop. (Previously, the .xlsx file was created but never actually opened so the user saw nothing happen)
11. Problem with not being able to use Excel 2007 on the report server has been resolved.
12. Problem with referring to 64bit columns via parameters in subtotal header/footer sections is resolved.
13. Column names containing ‘#’ are now permitted in csv files
14. Problem with user being repeatedly prompted for parameters set with setprompt in the control sheet is resolved.
15. It is now possible to write output from Excelerator in .csv format. (If the workbook has more than one sheet only the first sheet will be written)
16. Problem with control rows and columns being removed from Excelerator reports setup for post-back (containing update_columns row) is resolved.
17. “Remove Control Information” option is removed from Excelerator.

2.20. CHANGES IN SERVICE UPDATE 2

1. Problem with not being able to open Excel or Word files from https urls or network file locations when the Excelerator/Worderator add-ins are installed is resolved.

2. Datasets are now persisted in binary format rather than xml format within .rerx files. Thus new .rerx files will contain _data.dat rather than _data.xml. (Old rerx files remain compatible). This has been done in order to avoid the reference to the schema from w3c.org which, in some circumstances actually causes an unnecessary web request which may even time out.
3. Problem with duplicate names in Excelerator Load>Select… is resolved. (problem was introduced in SU1)
4. Problem with “where not xxx” syntax in “query balance” in Excelerator is resolved.
5. New $$dbtype system parameter is available in ReSQL which can be used to determine the type of the database 1 = MS SQL Server, 2 = Oracle, 0 = other.
6. Excel and Word documents output from Smart Client / Agresso Desktop are no longer opened in separate instances of Excel/Word.
7. New .attachFile and .attachDocument commands are available in ReSQL which enable the user to control files and documents to be attached when distributing reports by email. (This offers exactly the same functionality as agr.AttachFile() and agr.AttachDocument() in non-rerx ARC reports)
8. Problem in Data Explorer where the data contains a binary object which is not an image is resolved.
9. Problems with “invalid procedure call” or “object reference” errors when sending emails are resolved.
10. Problems causing timeout of AgrxlCmd subprocess due to revocation check are resolved.
11. Problems with Excel not opening in batch mode because of non-existance of desktop folder are resolved.
12. Problem with “System.IO.IOException: The directory is not empty.” Occurring occasionally when splitting reports is resolved.
13. Problem with report output not being imported into the database on 64bit report server queues is resolved.
14. Problem with splitting reports on a column containing null values is resolved.
15. Problem with unloading Excelerator parameter rows containing merged cells is resolved.
16. Problem with sql and other strings ending with single character token is resolved.
17. Exception details written to the log file are improved in the case where the exception has occurred within the web-service.
18. Minor correction to conditions under which user is allowed to execute “query balance”.
19. The default outline level for new Excelerator reports reverts to the same as it was in 8.28.
20. Problem with textboxes in Word documents causing an error in Worderator is resolved.
21. Correction to treatment of historical Excelerator syntax using braces {}.
22. Field list is now properly populated in ARC Script Editor.
23. This release requires the Report Engine web service to be updated. It is not compatible with any previous version of the web-service.

2.21. CHANGES IN SERVICE UPDATE 3

1. Excelerator is now fully compatible with 64 bit installations of MS Excel.

2. Problem with continuation output files not being saved when maximum number of sheets in a workbook was exceeded is resolved.
3. Problem with not being able to update Excel AppSettings during installation is resolved.
4. Problem with formulas in cells formatted as text in subtotal headers and footers is resolved.
5. Problem with language being set incorrectly when logging in via the web-service is resolved. (Previously the returned language was always ‘EN’)
6. Problem with “where not xx*” and “where not xx null” combined with “query balance” in Excelerator is resolved.
7. Attempting to use “internal” Excelerator splitting within an xlsx file embedded in an rerx file now fails cleanly.
8. Excelerator now has a new option named “Keep Loaded Sheets Only” which has the effect of removing _control as well as any template sheets etc from the output .xlsx file. This is useful when splitting and distributing .xlsx files contained in .rerx files.
9. Problem with “Invalid characters in path name” when supplying parameter named xf_file to a server based Excelerator report is resolved.
10. Problem with Inserting references to controls into script in ARC is resolved
11. The splitting of address fields into separate lines for Worderator Mail Merge is reinstated to give exactly the same functionality as in 8.28.
12. A set of translation files are now reinstated and improved. This is currently available for the Office designers. The Office designers will obey the Office language set if a corresponding translation file exist. The translation text can be customized if necessary.

2.22. CHANGES IN SERVICE UPDATE 4

This is a small Service Update containing only fixes to the following issues discovered in Service Update 3.

1. The Excelerator and Worderator Addins are now compatible with Office 2016.

2. Problems with Null or blank data or any 16bit integer data preventing Post-Back from working are resolved.
3. Problem with the text in the ReSql editor becoming corrupted is resolved.
4. Some of the titles in the Excelerator Add-in have been corrected.
5. Exception is no longer thrown when importing a report against a screen in the U4BW Desktop.
6. Fixed installation problems and moved check for MS Excel version from installation to run time.
7. Problem with intermittent crashes when opening the Data Explorer is resolved.
8. Problem with using internal Excelerator splitting with “Keep Loaded Sheets = false” is resolved.
9. Size of log file is reduced when splitting reports.
10. New xs_report and xw_report parameters are available to allow you to choose which report file from within an report package is to be used at each stage. Note that xs_report can contain parameters eg.
    Xs_report=Invoice_<language>.repx
11. A new “Refresh Data” button is available on the Excelerator and Worderator Add-ins and a corresponding menu item is added in Report Studio.
12. Problem with incompatibility between the Excelerator Add-in and certain other Excel Add-ins (notably Adobe Acrobat) is resolved.
13. Problem with Word Mail Merge when data contained nulls in address columns is resolved.

3. INSTALLATION

Installation instructions for Unit4 Report Engine 9.5.

This release of Report Engine is compatible with Unit4 Business World Milestones 3, 4, 5, 6 and 7 on the latest Software Update. Note the Report Engine web-service requires IIS7 or higher as is required by Unit4 Business World Milestone 3. (The new WebAPI based web service is only compatible with MS 7)

Excelerator and Worderator clients require MS Office 2010, 2013 or 2016 on their latest Service Packs. Either 64 or 32bit versions of Office are acceptable. Note: Office 2019 is simply a wrap-up of all the continuous updates to Office 2016 made available by Microsoft for non-Office365 customers and is also acceptable.

MS Excel is no longer used in any circumstance on any server machine and so it should not be installed.

Important implications of installing Report Engine 9.5:

- This release of the web-service will require any installations of Report Engine on client machines also to be 9.2 or above. However, it is possible to install this web service side-by-side with an older version rather than replacing it.
- This release of the Web-API server will support clients on earlier releases except if they are using IDS in which case the clients must also be upgraded to this release.

3.1. OFFICE 365 COMPATIBILITY ISSUES

The existing Report Engine Office Addins (Excelerator and Worderator) are not compatible with Excel Online or Word Online. They are compatible with “click-once” installations of the Office applications downloaded from Office 365. The Add-ins will behave exactly as they do in any other installation of Office.

3.2. INSTALLATION FILES

The zip file contains a set of necessary installation files to successfully install ReportEngine on a client or server. This must be installed on ALL machines (client, business server and web servers) on which Report Engine is to be used.

- **UnitReportEngine9.msi**: The ReportEngine client/server components. It will check for having the correct version (or newer) of Microsoft.Net and Visual Studio 2010 Tools for Office Runtime (if Excel or Word is found). If Visual Studio 2010 Tools for Office Runtime has previously been installed, please make sure that you have version 10.0.50903 or later. If you have an earlier version, you will need to upgrade it by downloading the latest VSTO runtime redistributable. Otherwise, the AgressoReportEngine9.msi will not complete successfully.
- **ReportEngine.pdf**: The ReportEngine user manual.
- **InstallationNotes.txt**: These installation instructions.

Before installation:

- On Web Server machines running the Unit4 Business World web application and/or the Report Engine web-service the respective web-sites should be stopped and re-started after installation is complete.
- Any running instance of the Unit4 Business Server should be stopped. (Including all server queues)
- Any running instance of Unit4 Business World desktop should also be closed.

The client application (Excel and Word Add-ins, XtraReports Designer and ARC Designer) is always installed. The installation exe also has a step which interrogates the registry to find the installed location of Unit4 Business World and installs Report Engine in all of the following that are found on the machine:

1. The Report Server files in the corresponding %INSTALL_PATH%\bin folder.
2. Any centrally configured client bin folders on the machine.
3. The Report Engine web-service (if present)
4. The Unit4 Business World Web Application (if present)

This step is carried out by the program named AgrxlInstall.exe. It always creates a log file with a name like – AgrxlInstall_nnnnnn.log under the current TEMP\ReportEngine folder.

If, for any reason, your copy of Unit4 Business World was not installed in the normal way i.e. by the main installation (e.g. a web server machine which does not have an installed copy of Unit4 Business World). You will need to re-run AgrxlInstall from the command line supplying the path of the installation target as follows. DO NOT copy any of the files located under the “ServerFiles” folder by hand.

To install Report Engine only in the named Unit4 Business World bin folder:

```
AgrxlInstall "%INSTALL_PATH%\bin"
```

To install Report Engine only in the named Unit4 Business World Web folder:

```
AgrxlInstall "%INSTALL_PATH%\Business World"
```

To install Report Engine only in the named Unit4 Business Web Service:

```
AgrxlInstall "%INSTALL_PATH%\Web Services\Report Engine"
```

Note: `%INSTALL_PATH%` refers to the location of Unit4 Business World installation files.

3.3. ADDITIONAL STEP REQUIRED FOR WEB SERVICE

If this is a new installation, once you have run the installation you will need to go into the Unit4 Business World On! Management Console and “create” the Report Engine web-service in the normal way.

Some versions of the Management Console may refer to a test mode to check the installation of the Report Engine Web Service. This is unfortunate because this feature was removed for security reasons and the test will now always fail.

3.4. NOTES

Note 1:

This version of the Report Engine web service will ONLY support ReportEngine 9.2, 9.3, 9.4 and 9.5 clients so this implies that any client machines running previous versions connecting via the web-service will need to be upgraded this version.

Note 2:

In order to use Post Back over the web-service connection it is necessary to add the following key in the AppSettings section of the web service web.config file.

```
<add key="AllowPostBack" value="true" />
```

Note 3:

There is a bug in some versions of the Management Console which causes the Report Engine web service virtual folder to be marked for "Directory browsing". This option should be unchecked by hand.

Note 4:

The parameters ACT_CL_LEVEL and ACT_TG_LEVEL should be enabled by setting the value to 0 or greater. However, this is the default so no action is usually necessary.

Note 5:

There is no need to remove or uninstall any copy of Report Engine 8.x as 8.x and 9.x can co-exist without problem.

Note 6:

If interactions with the web-service are unusually large (> 4MegaBytes) or unusually long-running (> 90 seconds) then it may be necessary to edit the Report Engine web service web.config file and uncomment the httpRuntime section so that appropriate values can be set for the executionTimeout and maxRequestLength settings.

Note 7:

It is important to make sure that the "Check for publisher's certificate revocation" flag is NOT checked in Internet Explorer>Internet Options>Advanced (see below). Otherwise every time that a signed dll is loaded the operating system will try to talk to Verisign over the internet to validate that the certificate has not been revoked. Report Engine and the Unit4 Business World platform contain many dlls all of which are signed so this can introduce a very serious delay (up to several minutes) when the application is started.

This setting is held on a per-user basis and will need to be disabled for the Business Server and Web Service users as well as regular client users. The easiest way to achieve this is via the following command:

```
AgrxlInstall revocation /disable
```

Note 8:

It is possible to change the treatment of long running (timeout > 30s) reports within Unit4 Business World Web. By default such reports are submitted to be rendered in batch on the report server. However, if you add the following app setting to the Unit4 Business World web web.config file then such reports will be streamed to the user as a report package (rerx file) with the auto-run flag set. This is the pre-8.28 behaviour but, of course, is only useful if Report Engine is installed on all client machines.

```
<add key="ReportEngineLongAsRerx" value="true" />
```

Note 9:

When initiated from Information Browser all long running (timeout > 30s) reports will be delivered to the client machine as a report package (rerx file) irrespective of the value of “ReportEngineLongAsRerx”. Note also that the ability to render such reports on the report server queue mentioned above (Note 8) is deprecated. This is due to licensing restrictions if the installation is within Microsoft Azure.

Note 10:

After the installation of an update of Unit4 Buisiness World you should re-run agrxlinstall so as to re-copy the required UBW dlls into the Report Engine web service bin folder.

Note 11:

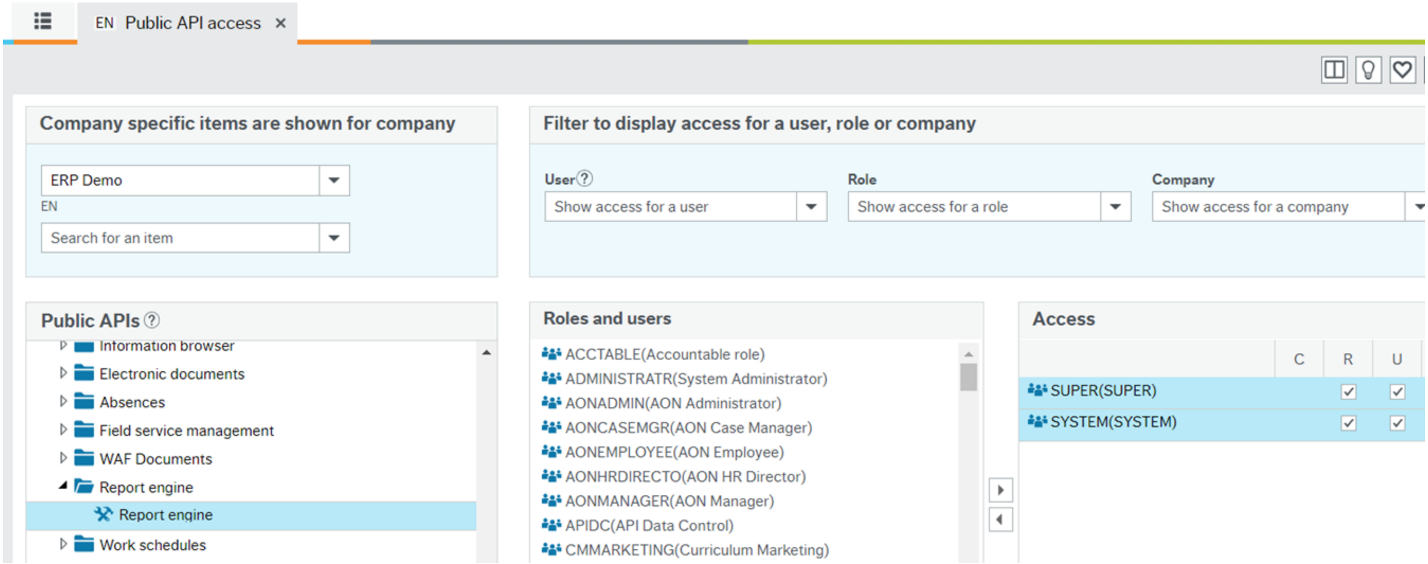
The IDS authentication is only supported by the "Unit4 Business World WebApi" connection and requires IDS 3.0 or above. Currently, only support for an implicit flow client is implemented. To be able to use the IDS connection the following ReportEngine IDS client values must be set:

Client name:	Report Engine Client
Flow:	Implicit
Access token type:	jwt
Access token lifetime:	86400
Allow access to all scopes:	no
Allowed CORS origins:	
Allowed scopes:	u4bw, openid, profile

Redirect URIs:	oob://localhost/reportengine, http://localhost/reportengine/
Post logout redirect URIs:	
Enabled:	yes
Require consent:	no

The access token can be used for SSO and should be set to an acceptable value (this does not depend on a client secret and should not be stored for a long period of time). The access token is encrypted within the registry.

The following access to the Report engine Web API must also be granted…



Company specific items are shown for company

ERP Demo

EN

Search for an item

Filter to display access for a user, role or company

User

Show access for a user

Role

Show access for a role

Company

Show access for a company

Public APIs

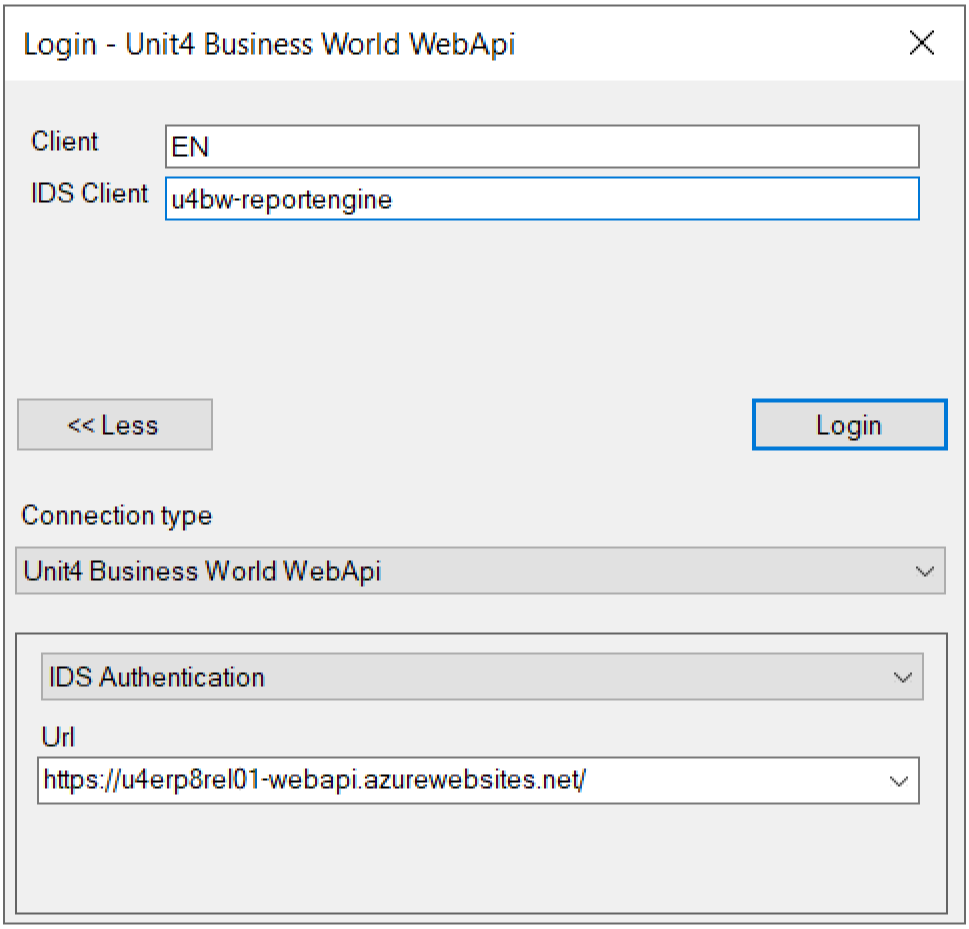
- Information browser
- Electronic documents
- Absences
- Field service management
- WAF Documents
- Report engine
  - Report engine
- Work schedules

Roles and users

- ACCTABLE(Accountable role)
- ADMINISTRATR(System Administrator)
- AONADMIN(AON Administrator)
- AONCASEMGR(AON Case Manager)
- AONEMPLOYEE(AON Employee)
- AONHRDIRECTO(AON HR Director)
- AONMANAGER(AON Manager)
- APIDC(API Data Control)
- CMMARKETING(Curriculum Marketing)

Access

	C	R	U
SUPER(SUPER)		☑	☑
SYSTEM(SYSTEM)		☑	☑



Login - Unit4 Business World WebApi

Client: EN

IDS Client: u4bw-reportengine

Connection type: Unit4 Business World WebApi

IDS Authentication

Url: https://u4erp8rel01-webapi.azurewebsites.net/

ReportEngine tries to get the IDS configuration using the WebApi Discovery endpoint (…/discovery)

Then, if not successful, it will try to use any valid database connection and request the System Parameters:

`IDS_AUTHORITY`

`IDS_TENANT_ID`

4. EXCELERATOR

Unit4 Excelerator is a report writer primarily aimed at producing the month-end management reports from Unit4 Business World. It achieves this by downloading data from the Unit4 Business World database into Microsoft Excel™ which is used to format and present the information. Functionally it is more similar to the traditional GL report writers that existed in some of the older accounting packages on the market than it is to the SQL based reporting tools (like AGRESSO Report Writer, AGRESSO Report Creator or Crystal Reports) that are commonplace in the market today.

The functional difference is very important but perhaps difficult to understand until one has actually been presented with the problem of producing the sort of balance sheet or P&L analysis report that is usually required for the month end management report pack.

Tools like AGRESSO Report Creator, Crystal Reports, AGRESSO Report Writer, MS ACCESS, Business Objects, etc are all SQL based (irrespective of whether you have to write the SQL yourself or if it is generated for you). That is to say that they are all based on the premise that you wish to print one line, group of lines or page on the report for each row returned from an SQL SELECT statement. This is not the case with Excelerator. Excelerator is based on the premise that you wish to specify that on a *particular* line in the report you want to show data summarized or pertaining to a particular range of accounts (or some other attribute(s)). This is exactly the case that occurs in the management report pack.

Unit4 Excelerator addresses the following sorts of issue that arise when producing a P&L analysis or similar report. It does this either directly within the product or by leaning on standard features of Microsoft Excel™.

- Asymmetric totalling (e.g. high level summary of income, detailed breakdown of expenses)
- Sign reversal on a line by line basis (e.g. income reversed, expenditure not reversed)
- Selection of data based on more than one attribute (for example account and sub-account)
- Data from a number of different sources (e.g. head count for a division interspersed between rows of financial data)
- Footnotes showing the detail that goes into a particular line at a later point in the report
- Calculated rows (e.g. % profit as well as sums, for example total profit).
- Calculated columns showing row value as a percentage of the total value.
- Cross tabs (e.g. P&L by cost centre)
- Cross tabs by user specified ranges (e.g. revenue by specified ranges of periods)

A very important point to note is that Unit4 Excelerator is not in any sense a substitute for a row based enquiry tool or report writer. For example it does not address ad-hoc enquiries like the AGRESSO Browser or balance table enquiries. Nor is it the best tool for laying out the printed page for printing on pre-printed stationary (obvious examples of this would be invoices, statements, cheques and remittances, etc).

4.1. CONTROL SHEET (_CONTROL)

Then control sheet (_control) controls the loading of the rest of the worksheets in the workbook. It allows you to define parameters and should contain a list of the worksheets to be loaded. Like the worksheets to be loaded, processing of the _control sheet is determined by the contents of column *A*.

The presence of the _control sheet is not mandatory for simple reports.

A template _control sheet can be created from dialog: Unit4 Excelerator, Options, Tools, Make Example Control Sheet.

Control Sheet Row Types:

4.1.1. Comment

Column *A*

*	Any row with an asterix (*) as the first non-space character in column *A* will be ignored. As will any row with column *A* left blank.

4.1.2. Sheet

Column A	**Column B**	**Column C**	**Column D**
Sheet[,noblanks][,nooverwrite] [itteration]	sheet name	template name	parameters

noblanks Indicates that sheets in which no query returns any data should be skipped.

nooverwrite Indicates that sheets in which output_dir \ output_file points to an existing file should be skipped.

sheet name This is the name of sheet to be processed.

itteration If this is specified, the sheet will be loaded from a template multiple times in which case *template name* is mandatory and *sheet name* should contain a parameter.

Itteration can be:

attribute *attribute client*

which will cause the sheet to be loaded once for each value of the attribute *attribute* that is defined for the Unit4 Business World client (company) *client*.

Excel *sheetname*

This will cause the sheet to be loaded once for each row in the specified Excel sheet (which must be in the same workbook). The format of such a spread sheet is the same as when a spreadsheet is used as the source for a query. This includes column names in row 1, data in row 2, onwards.

A valid SQL SELECT sentence which will cause the sheet to be loaded once for each row returned from the select sentence with local parameters set for each column returned from the select sentence.

template name This is the name of template sheet from which sheet is to be produced. If left blank, it is assumed that the sheet named in column 2 already exists.

parameters These are local parameters to be applied to this sheet with a format like:

LocalParameter1=Value1, LocalParameter2=Value2, …

Values used in both *parameters* and *insert strings* may themselves refer to earlier defined local or global parameters simply by referring to the parameter enclosed in angle brackets. For example (in *insert strings*):-

B2=Period <period>

meaning that the value of the parameter *Period* should be inserted into cell B2 prefixed by the word **Period**.

4.1.3. Group

Causes all sheets specified within the group to be loaded for each row returned by *iteration*

Column A	Column B	Column C	Column D	Column E
group [,noblanks][,nooverwrite] *iteration*	group name		parameters	
sheet	sheet name	template name		Insert strings
sheet	sheet name	template name		Insert strings
sheet	sheet name	template name		Insert strings
…				
endgroup				

noblanks indicates that sheets in which no query returns any data should be skipped

group name is a name for the group that will appear in the "Select Sheets dialog.

4.1.4. Break

A break row is always used in conjunction with a Group row and causes the sheets within the group to be loaded for each value of *column* encountered in the data rather than for each row in the data. If the Query control syntax is used within a report sheet within the group then the data for the current value of column will be available.

Column A	**Column B**	**Column C**	**Column D**	**Column E**
group [,noblanks] *iteration*	group name		parameters	
break *column*				
sheet	sheet name	template name		Insert strings
sheet	sheet name	template name		Insert strings
sheet	sheet name	template name		Insert strings
…				
endgroup				

4.1.5. Mandatory

Column A	Column B
mandatory	parameter

Causes an error if *parameter* is non-existent or is equal to a zero length string. This is useful in combination with setprompt to prevent the situation where a very long running query would otherwise be generated.

4.1.6. Declare

Column A	Column B	Column C	Column D
declare	parameter	type	Default value

Declares that *parameter* is of type *type* and allows you to specify a *default value*.

Valid types are: date, float, integer or string

If the parameter already exists (e.g. passed in from Unit4 Business World report ordering screen) then the default value will not be used, the existing value will be kept.

Default Value may itself contain references to previously defined parameters. Here as elsewhere this is done by prefixing the name of the parameter with a less than sign < and suffixing it with a greater than sign >. Example: If you have a parameter named *period* then you would refer to it as **<period>**.

4.1.7. **Set**

Column A	Column B	Column C
set	parameter	value

sets *parameter* to given *value.*

Value may itself contain references to previously defined parameters. Here as elsewhere this is done by prefixing the name of the parameter with a less than sign < and suffixing it with a greater than sign >. Example: If you have a parameter named *period* then you would refer to it as **<period>**.

4.1.8. **setdefault**

Depricated. Please use 'declare'

Column A	Column B	Column C
setdefault	parameter	value

If *parameter* is not already defined sets *parameter* to given *value.*

If *parameter* is already defined it has no effect.

Value may itself contain references to previously defined parameters. Here as elsewhere this is done by prefixing the name of the parameter with a less than sign < and suffixing it with a greater than sign >. Example: If you have a parameter named *period* then you would refer to it as **<period>**.

4.1.9. **setnum**

Column A	Column B	Column C
setnum	parameter	expression

sets *parameter* to the value of numeric *expression* which takes the form:

operand1 operator operand2

where operands 1 and 2 are numbers and operator is one of +, -, *, / or \.

(backslash (\) signifies integer divide, useful for obtaining year from period

199709 \ 100 = 1997) for example:

Column A	Column B	Column C
setnum	year	<period> \ 100

4.1.10. setperiod

Column A	Column B	Column C
setperiod	parameter	expression

sets *parameter* to the value of period *expression* which takes the form:

period1 operator period2 [, periods per year]

where period1 and period2 are valid Unit4 Business World accounting periods and operator is either

+ or **-**. The periods are added or subtracted modulo *periods per year*.

If *periods per year* is not specified it defaults to 12. For example:

Column A	Column B	Column C
setperiod	period1	<period> - 1
setperiod	period2	<period1> - 1

4.1.11. setsqlpattern

Column A	Column B	Column C
setsqlpattern	parameter	value

sets *parameter* to the value of *value* substituting SQL pattern matching characters for the corresponding regular expression characters.

Example: * is replaced with % which matches zero or more characters

and ? is replaced by _ which matches any single character.

This is useful when running as an Unit4 Business Worldreport or server process so that wildcards (usually a *) entered by the user in the report ordering window in the normal fashion can be converted to the corresponding characters to be used in an SQL statement.

For example:

Column A	Column B	Column C
setsqlpattern	costc	*

Results in the parameter *costc* being set to the value %.

4.1.12. setprompt

Column A	**Column B**	**Column C**
setprompt	parameter	value

prompts for a value for *parameter* using *value* as a default.

Value may itself contain references to previously defined parameters. Here as elsewhere this is done by prefixing the name of the parameter with a less than sign < and suffixing it with a greater than sign >.

Example: If you have a parameter named *period* then you would refer to it as **<period>.**

4.1.13. Setparameter

Column A	Column B
setparameter	*sql*

Causes the given sql sentence to be executed and will make the results available as parameters. Note. The SQL should return only one row, if more rows are returned this will be treated as an error that will cause the loading of the sheet to terminate.

4.2. REPORT SHEETS

Loading of data into the spreadsheet is controlled by the contents of Column A. The program will ignore any row in which column A is empty.

The format of the contents of the control column is as follows:

```
type, keyword1, keyword2 Value
```

Commas separate the row type and keywords and the value (if specified) is separated from the last keyword by one or more spaces.

The various row types are logically grouped according to their functionality and when they are processed.

4.2.1. **Data Rows**

Data Rows define where the data for the report is to come from:

4.2.1.1. Query balance

Format:

```
query [, query number] balance ID client [period] [language]
```

ID is the ID of the balance table within Unit4 Business World.

client is the Unit4 Business World client (or company).

period if given specifies that the data is to come from a periodic balance table for the specified accounting period.

language is the Unit4 Business World language code (defaults to EN for English).

For this type of query Unit4 Excelerator looks at all the columns that are referred to on the *columns* row as well as any *crosstab* row etc and constructs the required SQL to retrieve the data. Valid columns are:

- Attributes referred to by name.
- Attribute descriptions referred to by the name of the attribute with the suffix _text.
- For non periodic balance tables
- *Period*, the accounting period
- *Amounts,* see Amount column names
- For periodic balance tables
- Amounts as above suffixed by appropriate suffixes.

It is possible to refer by name to any Unit4 Business World attribute which is either posted on the balance table or related to one that is.

In a consolidated balance table the *client* attribute column can be referred to as *dim_client*. This is to avoid a conflict with the *client* column that always exists in the balance table and will refer to the head office client.

The data retrieved from the balance table can be controlled by the rows *relation*, *sort* and *where* which apply only to this type of query.

4.2.1.2. Query excel

Format:

```
query [, query number] excel worksheet_name|range_name
```

The worksheet given should contain column names in row 1 and data in the following rows (as would be returned by **Get External Data**).

4.2.1.3. query SELECT ...

Format:

```
query [, query number] SELECT ...
```

4.2.1.4. query AGRESSO

Format:

```
query [, query number] agresso [template name]
```

This causes Excelerator to take the data that it is given by Unit4 Business World, either the output of an *.arw report or the contents of a browser window.

If the template name is supplied, then the data comes from running the browser template via the Query Engine. This means user group security will be respected.

If working in immediate mode, then the data will be read from a sheet named _data that is written by Excelerator when a new report is generated from Unit4 Business World. This enables you to work on the design of the report off-line.

4.2.1.5. Query control

Format:

```
query [, query number] control
```

4.2.1.6. Where row

Applies to balance table and Excel type queries only.

Column *A* syntax:

```
where [, query number] column criteria
```

Causes the balance table query to return only rows where column matches criteria. Note *criteria* are to be specified as in code columns or cross tab rows and not as in an SQL WHERE clause.

4.2.1.7. NoZeros row

Applies to balance table type queries only.

Column *A* syntax:

```
nozeros [, query number]
```

Causes rows returned from a balance table query to be ignored if all value columns are zero.

4.2.1.8. Sort row

Applies to Balance Table, AGRESSO and Excel type queries only.

Column *A* syntax:

```
sort [, query number] column [desc] [, [column [desc]], … ]
```

Causes the rows returned from the balance table query to be sorted by the specified columns. The **desc** keyword can be used to specify that the sort should be in descending order.

4.2.1.9. Datasource row

Applies to SQL type queries only.

Column *A* syntax:

```
datasource [, query number] datasource
```

Specifies that this SQL query should be run against the specified secondary data source rather than the current primary data source.

Note

- All secondary datasource connections are made using and ODBC or DataCache connection
- The specified data source must be recognized by Excelerator and, if running on the report server, require no user interaction (for example must not prompt for a password) otherwise an error will occur.
- If the primary datasource is also of type ADO then this is still defined by the parameter *datasource*.

Example:

query select shipperid, companyname, phone from shippers order by shipperid			
datasource northwind			
columns	text shipperid	text companyname	text phone
detail			

This will open a connection to the **northwind** data source (A sample database shipped with MS Access and SQL Server) and run the specified query. Any number of queries such as this could appear on the same sheet or workbook as other queries running against the main Unit4 Business World database.

4.2.1.10. Relation row

Applies to balance table type queries only.

```
Column A syntax:

relation [, query number] column1 , column2 [, column3 …]
```

Tells Unit4 Excelerator that column1 is a relation on column2 which is a relation on column3 etc. The last specified column must be a posted attribute on the balance table or a related attribute that has been defined by an earlier relation row.

4.2.1.11. Tree row

Applies to balance table type queries only.

```
Column A syntax:

tree [, query number] tree name
```

Tells Excelerator that the named tree should be included in a balance table query. The columns named in the tree can then be used in the query simply by naming them.

Note:
If the tree name contains commas then it should be surrounded by single quotes. For example: tree account, costc and project tree.

4.2.1.12. SQL rows

```
Column A syntax:

sql [fragment of sql sentence]
```

This provides a way of entering an SQL statement, 256 characters long to be used in a subsequent query row. All the values against all SQL lines are concatenated together and used in the next query line. For example:

```
sql SELECT dim1, amount
sql FROM aglaggadp_<period>
sql WHERE client = '<client>'
query
```

4.2.1.13. SetParameter row

```
Column A syntax:

setparameter [sql sentence]
```

Causes the given SQL sentence to be executed and will make the results available as parameters.

Note The SQL should return only one row, if more rows are returned this will be treated as an error that will cause the loading of the sheet to terminate.

4.2.1.14. Set row

Column *A* syntax:

```
set parameter=value

setnum parameter=expression

setperiod parameter=expression

setdefault parameter=expression

setsqlpattern parameter=expression

setprompt parameter=default
```

Sets the specified parameter to the specified value. Parameters defined in this way within the report sheet are local and active only the current sheet. For a description of valid expressions please see help on corresponding _control sheet rows.

4.2.2. Placement Rows

Placement Rows define where the data from the query is to be put within the Excel worksheet

4.2.2.1. Columns row

```
Column A syntax:

columns [, query number]
```

The rest of the cells in the row should contain the names of the columns returned from the query to be inserted into *summary*, *group* or *detail* rows.

Syntax for other columns:

```
code column
text column
concat column
value column
default value
column
```

If the name of the column is prefixed by the word **code** then this is taken to be a code that will be constrained to the values specified in the individual summary or detail rows. Code columns are hidden in the processed spreadsheet.

If the name of the column is prefixed by the word **text** then this column will be treated as text even if the contents and data type are numeric (so that sign reversal will not be applied).

If the name of the column is prefixed by the word **concat** then this column will be treated in the same way as if it were prefixed with **text** except that strings will be concatenated together from all rows matching the criteria in summary rows (rather than returning the first non-blank value). The default delimiter is a single space but this can be overridden by specifying a delimiter (zero or more characters) **concat**':' **project**.

The "default" keyword signifies that the corresponding value should be inserted into cells in this column in all active rows. This is very useful for columns into which the user is going to be asked to enter data.

If the name of the column is not prefixed or prefixed by the word **value** then this is a value that will be summed etc.

For balance table type queries each *column* must either be the name of an amount column on the balance table, or the name of a posted or related attribute. Attribute names may be suffixed with _*tex* meaning that the description of the attribute should be shown rather than the attribute value itself.

For backwards compatibility, if a *columns* row is not specified for a given query then the column definitions will be taken from the rest or the cells on the *query* row.

4.2.2.2. Crosstab row

This row allows the setup of a crosstab.

Column *A* syntax:

```
crosstab [,query number] [column]
```

Value column syntax:

```
[criteria]          see Criteria
```

column is the name of a column in the query that is used to select records on a column by column basis. In value columns of the query the selection criteria for the column is inserted.

Example:

query..	amount	amount	amount	amount	amount
crosstab period	0000-9605	9606	9607	9608-9999	

This will cause the amount for the specified periods to be entered in each column. The last column where no criteria is entered will show the total amount for all periods.

It is possible to have more than one crosstab row per query in which case there is an implied AND of the criteria on the 2 (or more) crosstab rows

4.2.2.3. Round row

Allows you to specify the level of rounding that should be applied either to all columns or to particular columns. If no number of decimals is specified against a particular column then the number of decimals supplied in column *A* is used, if no number is supplied in column *A* then the default is zero. This means that data will be rounded to the nearest whole number.

Note. This rounding is applied as the data is inserted into Excel so that totals shown are the exact sum of the displayed figures.

Column *A* syntax:

```
round [,query number] [number of decimals]
```

Value column syntax:

```
[number of decimals]
```

4.2.2.4. Unit row

Allows you to specify the base unit for display of the data so that the data can be shown in thousands for example. If no unit is specified against a particular column then unit supplied in column *A* is used. Effectively this row causes all the data to be divided by the number given as unit.

Column *A* syntax:

```
unit [,query number] unit
```

Value column syntax:

```
[unit]
```

4.2.2.5. ColumnSign row

You can specify sign reversal on a column by column basis.

Column *A* syntax:

```
columnsign [,query number]
```

Value column syntax:

```
[flag]
```

flag should be set to blank, **n**, or **no** to indicate that sign reversal should not take place for this column. Set it to **y** or **yes** to cause the column to have sign reversal applied.

4.2.2.6. Summary rows

Column *A* syntax:

```
{summary | sum} [, query number] [,hidden] [,{sign | -}]
```

Code column syntax:

```
[criteria]          see Criteria
```

This row will be loaded with the sums of all the specified columns for all rows in the underlying query which satisfy criteria specified in the code column(s).

If the hidden keyword is specified then the row will always be hidden in the loaded sheet irrespective of the setting of the option **Zero Suppress**.

If the **sign** or **-** keyword is specified then sign reversal will be applied to data loaded by this row.

4.2.2.7. Subtotal row

Column *A* syntax:

```
    subtotal [, query number] [,outline[n]] [column1 [, column2 [,…]]]
```

This row type provides the classic break-pointed – subtotalled report functionality similar to that found in Crystal or ARW.

The modifier Outline modifier causes Excelerator to create an Excel Outline Grouping corresponding to the loaded data. The initial level of the outline that is displayed can be set by suffixing the outline keyword with the initial level. For instance, outline1 causes the structure to be collapsed to the top level. The default is **level 2** if no level is specified.

4.2.2.8. Header row

Column *A* syntax:

```
    header [, query number] [,page] [,merge] [column]
```

This row type is valid only in conjunction with a Subtotal row and defines a row to be inserted before every new value of *column* encountered in the data.

The modifier Page causes a page break to be inserted before the header row in the loaded report.

The modifier Merge causes the header to be merged with the first row of data.

The parameter substitution automatically occurs in header rows. The *$sum* etc keywords may also be used in the header row (see description in Footer row) to give a sum of the data that follows.

Header rows may be followed by one or more Continuation rows

4.2.2.9. Footer Row

Column *A* syntax:

```
    footer [, query number] [column]
```

footer rows may be followed by one or more Continuation rows

This row type is valid only in conjunction with a Subtotal row and defines a row to be inserted after every time the value of *column* changes in the data.

The parameter substitution automatically occurs in footer rows.

By default an Excel SUBTOTAL function is inserted beneath every value column.

If one of the following keywords is placed in a cell in the footer row (or any following continuation rows) then an Excel SUBTOTAL of the corresponding type will be inserted.

```
$average

$count

$counta

$max

$min

$product

$stdev

$stdevp

$sum

$var

$varp
```

4.2.2.10. Continuation row

Column *A* syntax:

```
+
```

a single **+** character in column *A* signifies a continuation row.

Continuation rows are valid only following Header rows, Footer rows or other continuation rows.

4.2.2.11. Detail row

Column *A* syntax:

```
detail | det [, query number] [,{sign | - }] [, above|below]
```

Code column syntax:

```
[criteria]          see Criteria
```

This will cause a row to be inserted into the spreadsheet for each row in the underlying query that satisfies the criteria specified in the code column(s).

The “above” or “below” modifiers allow you to specify if rows will be inserted above or below the original row.

If the **sign** or **-** keyword is specified then sign reversal will be applied to data loaded by this row.

4.2.2.12. Group row

Column *A* syntax:

```
group [, query number] ] [,{sign | - }][, above|below] column [, column]
```

Code column syntax:

```
[criteria]         see Criteria
```

This will cause a row to be inserted into the spreadsheet for each unique value of the specified column. Thus, it would have the same effect as specifying a summary row for each existing value of the specified column that meets the criteria specified in the code columns

The “above” or “below” modifiers allow you to specify if rows will be inserted above or below the original row.

If the **sign** or - keyword is specified then sign reversal will be applied to data loaded by this row.

If the group row is contained within a subtotal structure then the columns list on the Subtotal row is implicitly pre-pended to the columns list on the group row unless they are already explicitly included.

4.2.2.13. Range row

Column *A* syntax:

```
range [,query number]
```

Code column syntax:

```
[criteria]         see Criteria
```

This simply allows specification of criteria in the code column(s) to be OR'ed with the criteria in the next summary or detail row. Multiple range rows may be used.

4.2.2.14. Break row

Column *A* syntax:

```
break [,query number] column [sheetname]
```

A break row causes a separate output sheet named *sheetname* to be created for each value of *column* encountered within the data of the query. Note *sheetname* should contain a parameter that will differ for each value of *column* (usually *column* itself) for example to produce a profit and loss report for each division:-

break division P&L <division>

If *sheetname* is omitted then a default name is generated for the output sheets.

Within the output sheets the data in the query containing the break will be constrained to include only those rows for the current value of the break column. So that in our example P&L 42 will only see data for division 42.

The use of a break row within a report sheet is functionally very similar to using a break row within a group on the _control sheet but is more concise.

Only one break row is allowed per sheet.

Break rows are not allowed within a sheet that is already within a group on the _control sheet

4.2.3. **Cosmetic rows**

4.2.3.1. Parameter row

Column *A* syntax:

```
parameter
```

Has no effect except to cause parameters to be substituted in cells in this row

4.2.3.2. PreParameter row

Column *A* syntax:

```
preparameter
```

Has the same effect as a parameter row i.e. that all parameters in all cells in this row will be substituted. The difference is that preparameter rows are processed before anything else and after they are processed the sheet is recalculated and processing of the sheet recommences. This is so that the value of a parameter can be made available to be used in Excel Formulae early enough to affect the subsequent processing of the sheet.

4.2.3.3. Comment rows

Any row with an * as the first non-blank character in column *A* is treated and a comment. Such rows are hidden when the sheet is loaded (un-hidden when the sheet is unloaded) but otherwise have no effect.

4.2.3.4. Hide row

Column *A* syntax:

```
hide
```

Any columns containing anything other than blank or "0" in this row will be hidden after the sheet has been loaded. This may be parameterized in the normal way so that columns may be hidden or left visible according to the current values of parameters or data fields.

If the column contains “delete” then it will be deleted rather than just hidden.

Note: there may only be a single "hide" row for the whole sheet as the entire columns will be hidden for the entire sheet not for any particular query.

4.3. SQL

Anywhere where it is valid to enter an SQL sentence the following rules apply:

The SQL sentence must always return a recordset. This means it must usually be a SELECT statement.

You can use the portable SQL functions provided by Report Engine that will function whatever the database connection type against both SQL Server and Oracle. These functions all begin with a "$" and are documented separately.

The SQL sentence may be prefixed by DB (i.e. DBSELECT ….) to prevent any pre-parsing of the SQL by Unit4 Business World.

The SQL sentence must begin with SELECT or be prefixed by the EXEC keyword. The EXEC keyword exists to allow the possibility of running a stored procedure but this procedure must return a recordset. The EXEC keyword is only valid when running as a report or server process on the Unit4 Business World report server, it will cause an error if running in immediate mode or from the Unit4 Business World Client. This is a deliberate restriction to avoid the possibility of updating the database.

4.4. CRITERIA

Anywhere where criteria are specified the format is as follows:

```
[not] expression [;expression [;expression …]]
```

where semicolons represent a logical OR and the not keyword inverts the sense of the entire criteria string.

each *expression* can have one of the following forms:

a simple value

value

a range of values

```
value1-value2
```

to refer to a simple value containing a hyphen it is possible to quote values like this:

```
"aaa-bbb"
```

an inequality

```
< value       meaning less than value

> value       meaning greater than value

<= value      meaning less than or equal to value

>= value      meaning greater than or equal to value
```

an expression containing wildcards

```
*             matches any string of zero or more characters

?             matches any single character

#             matches any numeric character (0-9)

[list]        matches any single character from list

[!list]             matches any single character not from list

[a-c]         matches any single character in range a to c

[a-cA-C]      matches any single character in range a to c or in range A to C
```

the keyword **blank** to mean all rows for which the specified column is blank (zero length string)

the keyword **null** to mean all rows for which the specified column is null

the keyword **else** to mean all rows that have not yet been included in any other selection

the keywords **ref=n** or **ref>n** or **ref<n** to mean all rows referenced exactly **n** times, more than **n** times or less then **n** times. (**ref=0** is equivalent to **else**).

an empty string or * to mean all rows.

4.5. OPTIONS

These are the options:

4.5.1. Zero Suppress

Causes summary rows where no matching rows are found in the data to be hidden.

4.5.2. Optimized balance table query for Oracle

Causes Excelerator to generate the balance table SQL in a manner that is optimized for Oracle. (With recent releases of Oracle this no longer makes much difference)

4.5.3. Protect loaded sheets

Causes sheets loaded by Excelerator to be protected. If a parameter named *Password* exists this will be applied, otherwise the sheet will be protected without a password. Please note that this option is deprecated in favour of much better functionality. If the input sheet is protected then the output sheet will also be protected with the same protection options.

4.5.4. Remove printed / mailed sheets

Causes sheets that have been printed, emailed or saved to a separate worksheet to be deleted from the original workbook. Useful to prevent the workbook from becoming very large and creating a performance problem.

4.5.5. Query timeout

Applies the specified timeout (in seconds) to the database connection. If a value of 0 or a negative number is given then no timeout will be active.

Note:
This is not respected by all database/middleware combinations.

4.6. OUTPUT SPLIT

The output of Excelerator reports can be split into separate files (workbooks) and either emailed to separate recipients or saved in separate locations (or both). The output can also be saved as HTML (requires Excel 2000 or higher) or MHTML (requires Excel 2002 or higher). It is perfectly possible to request that the output is split to Email and to Documents.

Excelerator buffers the loaded sheets so that all sheets loaded sequentially with the same values of **email_to** and **email_subject** will be written into a single workbook. The Group feature on the control sheet can be used to control of the order of loading of sheets, so, for example all worksheets for a given cost centre could be mailed in a single mail to the cost centre manager.

This functionality is triggered by the presence of parameters named *email_to* and *output_file* and controlled by these and the other parameters listed below:

4.6.1. output_file

The name of the file where the loaded worksheet is to be saved. The type of file created depends on the file extension given in this parameter:

- If the file name has no extension or an extension of *.xls then an ordinary xls file will be created.
- If the file name has an extension of *.htm then an HTML file will be created. (Note. Under some conditions Excel will choose to create several secondary files as well as the primary file containing links to the secondary files.)
- If the file name has an extension of *.mht then a Web Archive (MHTML) file will be created. This is usually better than an HTML file in that it always creates a single.

4.6.2. output_dir

The directory into which the file is to be written. If omitted, and output_file is not a fully qualified pathname then the output file is written into the users temp directory.

4.6.3. output_title

The title property to be applied to the output file.

4.6.4. output_author

The author property to be applied to the output file.

4.6.5. output_subject

The subject property to be applied to the output file.

4.6.6. output_password

If password to be applied to the output file. If not present then no protection is applied.

Note:
This is not the same as the **password** parameter that is used in conjunction with the *Protect output* sheets Option which applies password protection to individual sheets rather than to the whole workbook (xls file).

4.6.7. email_to

The email recipient for the loaded worksheet. Or semicolon delimited list of email recipients.

4.6.8. email_cc

Email CC recipient for the loaded worksheet. Or semicolon delimited list of email cc recipients. (only when using Unit4 Business World Mail)

4.6.9. email_reserve

A single email recipient to use in the event of failure to send to the main email recipient.

4.6.10. email_subject

The subject field for the email.

4.6.11. email_text

The text to be included in the body of the message. (only when using Unit4 Business World Mail)

4.7. PERFORMANCE

Performance is not generally a problem in Excelerator reports bringing back a few thousand rows. In such cases the rest of this section is not relevant. However, in cases where a few hundred thousand rows are to be loaded these issues can become important.

The new "ASPOSE" method of manipulating .xlsx files without involving MS Excel is intrinsically faster than the previous methods involving MS Excel. Users should see a considerable (two to three times) improvement with this release. Many of the considerations that used to be listed in this section are no longer relevant.

However, whatever version of Report Engine is in use, there are a number of factors that affect the speed at which an Excelerator report will run. Some of these are intuitive, others are much less obvious. These are discussed below.

4.7.1. Data Extraction

The data extraction process is considered separately. Clearly a given SQL statement or Browser Template will execute in the same time irrespective of changes introduced in 9.x. However, using ReSQL it is often possible to avoid re-running similar queries thus saving a considerable amount of time.

4.7.2. Data Transfer

The transfer of data from Excelerator into Excel is the most expensive part of the processing. It goes without saying that you should never bring back more data into your .xlsx file than you need. This remains true but it is exactly at this point that the improvement using ASPOSE is greatest.

4.7.3. Deployment

Reports will now run at similar speeds on the report server as they do directly within the Excel Add-in.

4.7.4. Wildcard Comparators

It is best to avoid excessive use of wildcards in comparators (code columns and crosstab rows) as it takes much longer to match the data to the wild carded pattern than to compare with fixed values. (This is not usually a problem unless many such comparisons exist in the report)

4.7.5. Multi-row headers / footers

Copying header and footer template rows accounts for a significant fraction of the time taken to process "subtotal" structures. Adding continuation rows to headers or footers significantly increases this time. Thus it is best to avoid adding continuation rows that are nothing more than spacers to introduce a blank line between subtotals. (It is better simply to increase the height of a single row and set the alignment to "top").

4.7.6. Remove loaded rows / unload before save / import

It is perfectly possible to load an already loaded sheet. The result will always be correct but doing this demands that the sheet first be unloaded as part of the same process which may take a significant amount of time. It is therefore best practise to make sure that all template reports are unloaded before importing into the system or placing in "Customised Reports"

4.7.7. Conditional Formatting

The use of conditional formatting in rows that are to be copied during the "load" process can make a catastrophic difference to the time taken to load the sheet. Furthermore the performance hit goes as the square of the number of rows and so large reports are particularly badly hit. This is due to changes that were made in Excel itself and came in with Excel 2007. Excelerator puts a warning into the log file if it encounters such a situation. It is strongly recommended that this construct be avoided. In particular because the performance problem may not be noticed during development of the report against small volumes of data.

4.8. IMMEDIATE MODE

In this mode Unit4 Excelerator is invoked directly from within Excel via a special menu item named **Unit4 Excelerator** that appears on the Excel worksheets menu bar. This is a very useful mode when developing and testing reports (especially in conjunction with the ability to read data previously saved into a worksheet). If a connection to the database is required in this mode then this has to be done via the Report Engine web-service or an ODBC connection.

4.9. UNIT4 BUSINESS WORLD REPORT

Report Engine reports can be run directly from Unit4 Business World as a report process on the report server. In this case the database connection provided by Unit4 Business World will be used and no additional configuration of data sources is required. To do this either set up a user defined menu item as described below or simply select an output type of Excelerator, Report Creator in the report ordering window of an existing report. If no xls or rpx file yet exists for the chosen standard report, then one will be created.

To setup a user defined Report Engine report in Unit4 Business World:

Set up an Unit4 Business World menu item to invoke this program as a server process. In **User defined menus** enter a new row as follows:

CO	Company to which menu item applies (or * for all)
ID	Unique ID for this menu item
Bespoke	Checked
Func Type	Report
Function	Report name (e.g. XL01 – excelerator will look for a report file of this name)
Argument	Not required – leave blank
Menu	Menu ID (for example 00 for Personal Menu)
Program	Not required – leave blank
Menu Text	Text to appear in menu
Variant	0
#	Sort value to determine position in menu (e.g. 99 to make it appear at the bottom)

Set up an Unit4 Business World user defined report to set the parameters as needed (see below). For the new menu item, make an entry in **User defined reports**. Here the fields have their normal meaning except that the fields *width* and *E-mail* are ignored. Any parameters defined will be passed through to Excelerator and available as parameters as if defined by a set row in the _control sheet.

When Excelerator or Report Creator are invoked as an Unit4 Business World Report, Report Engine will look for an input Excel workbook, or Report Creator rpx, in the report input folders (Customised Reports etc).

The output file will be placed into "Report Results" and also added as a blob into the database so that is available from "Maintenance of Ordered Reports" within the Unit4 Business World Smart Client and Self-Service.

Progress and error messages will be written into the Unit4 Business World log file in the normal way.

Excel only: If an Unit4 Business World parameter named *sheet* is defined then its value will be interpreted as the name of a sheet within the workbook to be loaded. *sheet* may also be a semi-colon delimited list of sheets. If this parameter is not defined or left blank then all sheets will be loaded.

Other parameters defined in **User defined report** and set in the report ordering window will be available for use within the Unit4 Excelerator report.

The following standard parameters are always available

Client	The current Unit4 Business World company
Lg or Language	The users language
User_id	User ID
Report_name	The name of the report

4.10. UNIT4 BUSINESS WORLD CLIENT INTEGRATION

Excelerator or Report Creator can be run directly from within the Unit4 Business World windows client. In this case Excelerator will use the database connection provided by Unit4 Business World and no additional configuration of data sources is required.

4.11. APPENDIX

4.11.1. Amount types and their column names on balance tables

Transaction Type	Amount Type	Column Name
General Ledger	Amount	amount
General Ledger	Curr. Amount	cur_amount
General Ledger	Debit amount	debit_amount
General Ledger	Credit amount	credit_amount
General Ledger	Amount3	value_2
General Ledger	Number	number_1
General Ledger	Value	value_1
Budget	Budget	budget
Budget	Budget currency	budget_cur_amt
Budget	Rev. budget	budget2
Budget	Forecast	forecast
Budget	Budget number	bud_number
Budget	Budget qty	bud_value_1
Contract Accounting	Ordered contr.	cn_com_amount
Contract Accounting	Comm. Contract	cn_rest_com_amt
Grants	Assignm	grant_amount
Unauthorised invoices	Reg. Amount	ua_amount
Unauthorised invoices	Reg. Curr. amnt	ua_cur_amount

Transaction Type	Amount Type	Column Name
Purchase	Goods received	po_vow_amount
Purchase	Purch. not received	po_rest_com_amt
Purchase	Purch. ordered	po_com_amount
Sales Orders	Contr. margin	so_margin
Sales Orders	Goods dispatched	so_vow_amount
Sales Orders	Incompl. order	so_rest_order
Sales Orders	Invoiced	so_inv_amount
Sales Orders	Sales ordered	so_com_amount
Sales Orders	To invoice	so_rest_invoice
Approved budget	Approved budget	bt_budget
Budget adjustments	Budget adjustments	bt_rev_budget
Commitments	Commitment amount	cm_com_amount
Commitments	Commited amount currency	cm_com_curr
Pay schedule	Commitment amount	pc_com_amount
Pay schedule	Commited amount currency	pc_com_curr
WBS-budget	Original budget	cm_mce_amount

Transaction Type	Amount Type	Column Name
WBS-budget	Rev. budget	cm_cce_amount
WBS-budget	Forecast	cm_fct_amount
WBS-budget	UEC	cm_uec_amount
Approvals for expenditure (AFE)	Approved (AFE)	cm_afe_amount
Cash Accounting	Cash amount	cash_amount
Financial Plan	Amount	plan_amt
Pre-order	Requested	cm_pre_order
Schedule (VOW)	Schedule (VOW)	cm_vow_amount

4.11.2. Column name suffixes on periodic balance tables / data warehouses

window prompt	Column Name Suffix
Current period	_p
Last period	_pp
Current period last year	_ppy
Year to date	_ytd
Year to date last year	_pyytd
Forecast	_pfc
OB current period	_ppytd
Current alt. period	_pr
Alt.period last year	_pypr
Current year	_y
Total to date	_td

These are combined with the column names from the non-periodic balance table to give the column name on the periodic balance table, for instance the column name General Ledger, Amount, Year to date last year would be **amount_pyytd**.

5. XTRA REPORTS

The Report Engine integration with XtraReports is primarily intended to provide a mechanism to create reports that are either to be printed or to be delivered as pdf files where presentation and the exact layout of the report are paramount. A very important example of such reports are all the "Standard Outputs" from Unit4 Business World (Statements, Sales and Purchase Orders, Invoices, Reminder Letters etc). I.e. it replaces the old "ARC" reporter.

XtraReports is a component supplied by DevExpress and is actively supported by them. A lot of useful general information about this product is available from their web-site and support forum. It is not the intention here to repeat such material. Instead we concentrate on the touching points between XtraReports and the rest of Report Engine and where Report Engine explicitly adds value.

There are a number of sample report packages (".rerx" files) shipped with the installation of Report Engine which illustrate best practice for creating XtraReports.

5.1. DATA BINDING

In the context of Report Engine, XtraReports relies entirely on RESQL to extract the data from the database and present it as a .NET DataSet. It is important to understand the way in which the data from such a DataSet can be "bound" to controls within the report.

A Report has a DataSource property which, in the Report Engine context is always set to the DataSet provided by the RESQL. It also has a DataMember property which determines which table within the DataSet will be used as the driving data for the report.

A very important special case is that the DataMember may also be set to the name of a table in the DataSet prefixed with a "$". This will cause ReportEngine to create a new table including all columns from all parent tables as well as the named table itself. This is exactly equivalent to using "SELECT * FROM *EXPAND* main" in an ARC report.

Note that this de-normalization occurs "just in time" when the report is being rendered – the data is never persisted in this de-normalized form avoiding memory and performance issues that would be result in putting such a statement into the resql. Some extra processing time is involved but this is usually more than mitigated by the fact that the report no longer contains "Detail Reports" which, themselves, demand a lot of resource. Essentially, this means that Xtra Reports may be created using the simple/traditional model without incurring the unacceptable performance penalty that would have been incurred with previous versions. It is recommended that this feature be used in all new reports whenever the dataset has any structure/relations.

Also, very importantly, this new feature is used by the ARC to Xtra converter in order to generate an XtraReport which has exactly the same structure as the original ARC report and which will be much more easily understood. Usually, the converted report will just need Calculated Field and Formatting Rule expressions to be completed "by hand".

5.2. DETAIL REPORTS

Another way of making use of the relational structure of a DataSet is to use “Detail Reports”. The DataMember of a detail report is set to a DataRelation whose parent table must be the DataMember of the parent report and whose child table contains the data to be used by controls within the DetailReport. This, however, can result in a report which is difficult to understand. Use of the “$tablename” feature is almost always to be preferred but older reports predating this feature may still be encountered.

5.3. DUPLEX PRINTING AND WATERMARKS

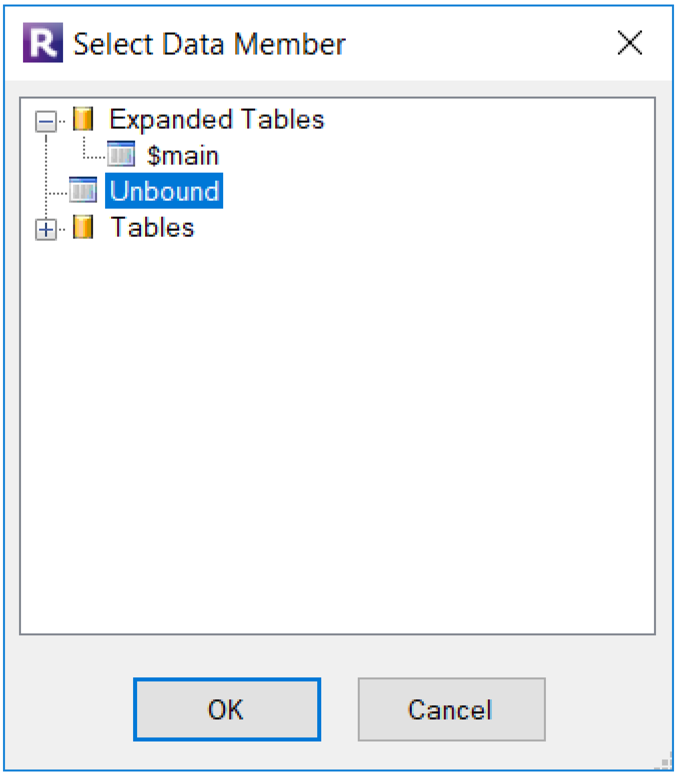
The ReportEngine XtraReports integration offers additional functionality in order to support duplex printing and variable watermarks. This is achieved using 3 concepts.

1) “Page Reports” - unbound single page reports contained in the .rerx file.
2) a special parameter named “xtra_watermark”
3) label controls with special names all beginning with “_watermark” or “_insert”

Note that to use this feature ReportEngine 9.4.4xx or higher is required at run-time on the machines onto which the report is to be deployed as well as on the development machine.

5.3.1. **Page Reports**

A “Page Report” is simply an unbound single-page report that is to be used to define the layout of a watermark or inserted page. It may refer to parameters but not to any other data. These can be created by selecting the “Unbound” option when creating a new XtraReport within Report Studio.



At run time the Page Report is rendered to create a single page that will be inserted into the main report or used as a watermark in the main report. However, if the Page Report itself has a watermark and the Page Report is being used to define a watermark, then the watermark from the Page Report is simply copied into the main report and any other contents of the Page Report is ignored.

5.3.2. xtra_watermark parameter

The “xtra_watermark” parameter simply allows the watermark to be applied to the whole report to be controlled from within ReSQL. The value given should simply be the name of the required Page Report (without the .repx extension).

For example:

```
.declare mode_flag string 'C'
.declare xtra_watermark string ''

.if '$?mode_flag' = 'C'
    .set xtra_watermark = 'Copy'
.endif
.if '$?mode_flag' = 'T'
    .set xtra_watermark = 'Test'
.endif
```

Note: It is generally more efficient to apply the watermark once in this way to the whole report than to apply the same watermark individually to each page. This is especially true if the watermark contains images.

5.3.3. **Special Controls**

Label Controls with names starting with "_watermark" or "_insert" are special controls recognized by ReportEngine and have special meaning to allow functionality required for duplex printing and applying watermarks. None of them will actually appear on the finished report (they will be highlighted in yellow by "Tools>Standardize Control Names").

Rules that apply to all special controls:

- The names of the special controls are case insensitive.
- If you need more than one instance of the same control in the report, you should suffix the second and subsequent instances with a simple integer. E.g. _watermark2, _insert3 etc.
- The "Text" property of the Label Control should be bound to a field that is driving the grouping of the report. E.g. if "_insertLast" is bound to "Customer Number" then a page will be inserted after the last page on which the control is printed for each customer. (Applies only to _insertFirst, _insertLast, _watermarkFirst, _watermarkLast and _duplex)
- The "Tag" property of the Label Control can be used to specify the name of a single page unbound report to be inserted or from which a watermark should be extracted. This can either be set to a constant value or bound to a field or parameter.

5.3.3.1. _watermark

This control will simply cause the watermark specified by the "Tag" property of this control to be applied to the page on which the control appears. Note: it would be meaningless for 2 _watermark controls to appear on the same page.

5.3.3.2. _watermarkLastOdd

If used this control must be used in the report such that it would appear on every page. This will cause the named watermark to appear on the last odd page of the group. This is useful for optical recognition marks that must appear on the front of the last sheet for a given recipient.

5.3.3.3. _watermarkLastIfOdd

This will cause the named watermark to be applied to the page if it is the last page in the group and it is an odd numbered page. Note this useful in the case where the control lies within a group header or group footer section marked "Repeat on Each Page".

5.3.3.4. _watermarkIfOdd

This will cause the named watermark to be applied to the page if it is an odd numbered page.

5.3.3.5. _watermarkFirst

This will cause the named watermark to be applied to the first page within the group.

5.3.3.6. _watermarkLast

This will cause the named watermark to be applied to the last page within the group.

5.3.3.7. _insertIfOdd

This control will cause a blank page or the template page named by the “Tag” property to be inserted after the current page if it is an odd numbered page. This is normally used within a group footer section and provides for the basic functionality required for duplex printing.

5.3.3.8. _insertLastIfOdd

This control will cause a blank page or the template page named by the “Tag” property to be inserted after the current page if it is the last page in the group and it is an odd numbered page. This must be used rather than _insertIfOdd if used within a group footer marked “Repeat on Each Page”

5.3.3.9. _insertFirst

This will cause a blank page or the template page named by the “Tag” property to be inserted after the first page within the group.

5.3.3.10. _insertLast

This will cause a blank page or the template page named by the “Tag” property to be inserted after the last page within the group.

5.3.3.11. _insert

This will cause a blank page or the template page named by the “Tag” property to be inserted after the page on which this control appears. Note: all other variants of “_insert” take precedence over this control.

5.3.3.12. _duplex

This will cause a blank page or the template page named by the “Tag” property to be inserted after the page on which this control appears if it is an odd numbered page. If used within a group footer marked “Repeat on Each Page” then this control should be bound to a the field driving the group footer and the blank page will only be processed for the last instance of the group footer.

6. REPORT CREATOR – ARC

Please Note that ARC is now deprecated. XtraReports should be used for any new work that is undertaken and if changes are required to an ARC report it should first be migrated to XtraReports. Please see separate document entitled “ARC to Xtra Migration Guide.pdf” which is shipped with Report Engine.

There are already (December 2019) a considerable number of known problems / limitations with ARC:

- Pure ARC reports (.rpx files) will not run on a 64bit Report Server queue.
- ARC reports embedded in “.rerx” files (aka NARC reports) will work on a 64bit queue but, when splitting, there is a very significant overhead (about 1 or 2 seconds per split) due to the fact that a separate (32bit) process is required for each split.
- ARC reports will not run on any web-server under any circumstances
- The pdf password protection / encryption is now very weak (can be broken in seconds by various tools which are freely available on the internet)
- Poor memory handling when processing large reports (especially if they contain images) combined with the fact that ARC is 32bit software causes these reports to run slowly and puts an upper limit on the maximum number of pages in the report output (typically 500 to 1000).
- The mechanism to export the results of the query in an ARW report into ARC is now unreliable as the self-registration mechanism no longer works in most cases so it relies on registry entries made at installation time which may well no longer be correct.
- VBScript is seen as a security risk by Anti-Virus software and so manual intervention to “whitelist” the processes in which ARC is running is often required.
- There are screen resolution issues in the ARC designer which usually lead to “fuzzy” text in the script editor.
- The VB6 runtime is required but is not always installed by default on modern installations of MS Windows.
- No support for QR codes.
- No support for Editable Fields.
- Some image files created by modern systems cannot be displayed by ARC. Such images do not cause an exception but simply result in a blank space in the report.
- It is no longer possible to create “.rec” cache files.

6.1. BASIC CONCEPTS

6.1.1. Controls

Controls are anything that is shown on, or inserted onto, the report surface. These in turn can be attached to fields of data from the results set, or they can contain the results of formulae, etc. They come in the form of, for example, Labels, Text Boxes and Lines.

Inserted on Report Surface

Define the Report Layout

Bound to Data Fields

6.1.2. **Data Fields**

These are the ways of actually getting information to the controls that we have already discussed. They come in the form of, for example, Fields, Parameters and Formulas.

The Underlying Data

Not just the Fields from the driving SQL

You can add a textbox control to your report, which is **bound** to a data field from the result data.

Note:

There are occasions when you would need to use an unbound field as well.

The **page** is displayed here to allow you to add, remove and alter the controls within the sections of the report. Every section on the report has a grey horizontal bar at the top of it. At the top of this bar, your cursor will change to a two-headed arrow, indicating that you can change the size of the section above. To effectively 'remove' a section from your report, you can simply drag the bottom line of the toolbar up, so that no part of the section remains visible. If you wish to temporarily 'hide' a section whilst you are, for example, comparing two other sections, you can simply click on the minus sign (**-**) in the top left-hand corner of the grey horizontal bar.

6.1.3. **File Types**

6.1.3.1. Report Definitions [.rpx]

Report Definitions are stored in files with the suffix **.rpx**.

These will generally be placed in the Customised Reports folder on the server.

6.1.3.2. Output files [.rdf]

The default report output format is written to files with the suffix **.rdf**.

This may be overridden by the xw_filetype/ xs_filetype parameters for any given report.

6.1.3.3. Canvas Files

Canvas files can be thought of as electronic pre-printed stationary allowing you to define the background to your report including, but not limited to, boxes. Additionally, it is possible to specify different canvas files on different page, or between the pages of a report output.

Canvases are files in their own right with the extension **.cnv**. They need to be stored next to the rpx file.

If you are using a single canvas then the default name would be the same as the rpx file. So with a standard Purchase Order the two files would be **po01.rpx** and **po01.cnv**.

6.1.4. **Page setup**

Page setup	It is in here that you set the margins for the report. The measurements in AGRESSO Report Creator are in twips, which is a printers' term. A twip is a twentieth of an inch point. There are 72 points to an inch.
Hence there are 20*72 = 1440 twips per inch.
As an inch is 2.54cm, there are approximately 567 twips per cm.
A4 = 21021*567mm by 297mm -> 11905 twips by 16837 twips

Printer setup	Here you choose or alter the paper size, width, height, orientation of the page. You can also change the settings for collation, duplex printing and paper bin

Grid settings	In addition to the Alignment toolbar, you can also choose to display the grid here and whether any new controls on the report will be aligned to the grid. You can also change the number of rows and columns in the grid here, as well as opting for the grid to be set in inches or centimetres

Styles	You can set new font styles here and alter existing ones

6.1.5. Page Setup Further Information

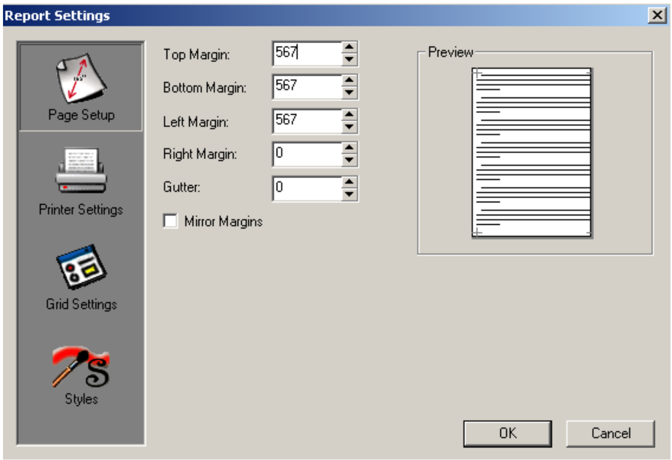
If you have a vertical red line on your **Preview** tab, this indicates that your page and print setup are not correctly completed and information falling to the right of this line is not within the printable area of the page. You need to resolve this otherwise you will get extra blank pages when you print the report.

For example, if you are working with A4 portrait paper and wish to have a one centimetre margin, you would make sure the following are set:

The ActiveReport property, **PrintWidth** needs to be set to **11338**

From the File menu, the Print Setup needs to have **A4** and **portrait** chosen

From Page Setup in the File menu, ensure that the left margin is set to **567** and right margin is set to **0**, and then click on the Printer settings button.



Top Margin: 567

Bottom Margin: 567

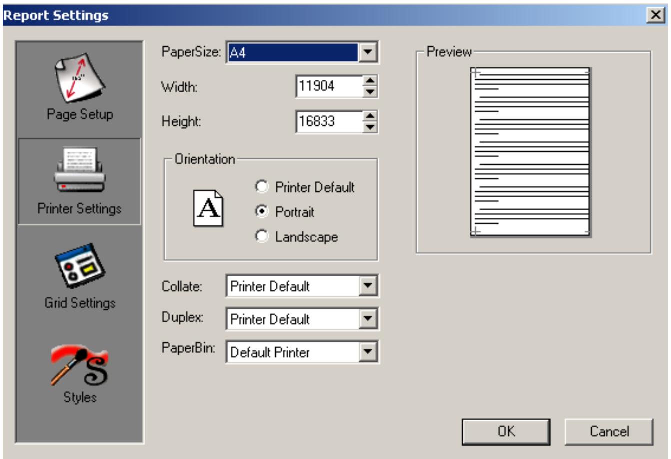
Left Margin: 567

Right Margin: 0

Gutter: 0

□ Mirror Margins

You then need to ensure that **A4** is chosen as the **size**, **portrait** is chosen as the orientation, the **width** is set to **11905** and the **height** to **16837**.



PaperSize: A4

Width: 11904

Height: 16833

Orientation

○ Printer Default
● Portrait
○ Landscape

Collate: Printer Default

Duplex: Printer Default

PaperBin: Default Printer

If you are working with A4 landscape paper and wish to have a one centimetre margin, you would make sure the following are set:

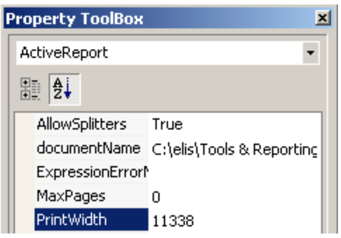
The ActiveReport property, **PrintWidth** needs to be set to **16269**

From the File menu, the Print Setup needs to have **A4** and **landscape** chosen

From Page Setup in the File menu, ensure that the left margin is set to **567** and right margin is set to **0**, and then click on the Printer settings button. You then need to ensure that **A4** is chosen as the **size**, **landscape** is chosen as the orientation, the **width** is set to **16837** and the **height** to **11905**

6.1.5.1. Print Width

The report itself has a print width that must be set. If you are wishing to set margins, then the PrintWidth that you could use will be the maximum for that page size, minus the margins.



AllowSplitters	True
documentName	C:\elis\Tools & Reporting
ExpressionError	
MaxPages	0
PrintWidth	11338

Once you have chosen a page size, the print width is set automatically.

6.1.6. Titles

With the standard reports you can often use **Titles** instead of **Labels**. These are the titles of the datafields as they are set up within one of the tables in Unit4 Business World. This may be of little use on a standard site, but if your company is multi-lingual, you would use **titles** to allow the report to be easily translated into another language, simply by who has requested the report, or by the language as set on a particular customer masterfile.

Sample

Cs04.rpx

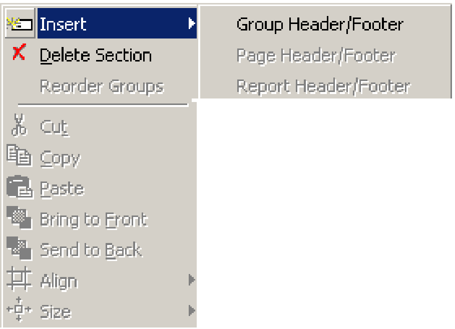
6.1.7. **Groups**

Groups are used extensively in reports. Any subtotalling that you may wish to do, will be done on a group. For example, if you wish to add a subtotal on your report, so that you can see what the total debt is that a customer has, you would first need to insert a group.

A group consists of a header and a footer – the header will come above the detail section and the footer will come below it.

Note

Using a group within AGRESSO Report Creator does not have any effect upon the order of the data. You would need to have already changed the order of the data from the ARW file. This is not covered on this course.



6.2. AGR OBJECT

The agr Object is available in the VB Script environment in Report Creator. It makes available a number of Properties and Methods that necessary when writing reports against the Unit4 Business World database.

6.2.1. Primary Properties and Methods

6.2.1.1. UseDataSetOnly

```
agr.UseDataSetOnly
```

Specifies that the report will read the data from the supplied dataset – No direct access to the database will be allowed. This is designed to be used in conjunction with a .resql file packaged in the same report package (.rerx file) as the report which will drive the extraction of the data from the database and the creation of the DataSet.

Note that the following statements are not required and illegal in a report specifying agr.UseDataSetOnly:-

```
agr.ParserType

agr.UseParameter

agr.UseTypedParameters

agr.UseTitle
```

6.2.1.2. ParserType

```
agr.ParserType = 1  ' 1=Native, 2=ASQL
```

Specifies the level of SQL parsing to be applied. This should be specified in all reports at the top of the OnDataInitialize() method. If not, ARC will revert to the historic behaviour which is that "Native" will be used when running from within the designer or from the Unit4 Business World client and "ASQL" will be used when running on the Unit4 Business World Report Server. This is almost never what you want.

AttachFile

```
    agr.AttachFile fileName
```

Specifies that the named file should be attached to the email when the report is distributed. This is a better alternative to using the xa_ parameters. This could be used to attach a file for each invoice line for example.

The arguments are as per agr.GetFile

6.2.1.3. DataSource

```
 agr.DataSouce = "my_data_source"
```

```
or
```

```
 agr.DataSource = ""
```

Specifies that the alternate (ODBC) datasource is to be used for all SQL (agr.SetQuery, agr.SelectInto and agr.GetRecordSet) until cancelled by another agr.DataSource.

agr.DataSource = "" means revert to the primary Unit4 Business World data source.

Note: This does not affect the Unit4 Business World (former Agresso) specific functions such as agr.GetDescription / agr.GetDocument which always work against the primary Unit4 Business World data source.

6.2.1.4. HasData

```
If agr.HasData Then…
```

returns True if the agr.SetQuery method has returned one or more rows and **False** otherwise.

Example:

In the OnFormat Event of the Report Header you might use the following code:

```
If agr.HasData Then
 rpt.lblReportTitle.Caption = "Resource List"
Else
 Rpt.lblReportTitle.Caption = "Resource List - No Data"
End If
```

Note: You must not use brackets on a function that takes no arguments.

```
agr.ParserType = 1  ' 1=Native, 2=ASQL, 3=DataSet
```

This should be specified in every report and is generated into all new reports.

Specifies the level of SQL parsing to be applied. If you do not specify this ARC will revert to the historic behaviour which is that "Native" will be used when running from within the designer or from the Unit4 Business World client and "ASQL" will be used when running on the Unit4 Business World Report Server. This is almost never what you want

```
agr.AttachDocument client, DocType, Key1, [Key2,] … [Key6]
```

Specifies that the Named document should be attached to the email when the report is distributed. This is a better alternative to using the xa_ parameters. This could be used to attach a document for each invoice line for example.

The arguments are as per agr.GetDocument

```
agr.AttachFile fileName
```

Specifies that the named file should be attached to the email when the report is distributed. This is a better alternative to using the xa_ parameters. This could be used to attach a file for each invoice line for example.

The arguments are as per agr.GetFile

```
     agr.DataSouce = "my_data_source"

    or

     agr.DataSource = ""
```

Specifies that the alternate (ODBC) datasource is to be used for all SQL (agr.SetQuery, agr.SelectInto and agr.GetRecordSet) until cancelled by another agr.DataSource.

agr.DataSource = "" means revert to the primary Unit4 Business World data source.

Note: This does not affect the Unit4 Business World (former Agresso) specific functions such as agr.GetDescription / agr.GetDocument which always work against the primary Unit4 Business World data source.

6.2.1.5. UseParameter / Parameter / UseTypedParameters

```
agr.UseTypedParameters
```

tells the system to use typed parameters. This call should be used in all new reports and is included in the code that is generated automatically. If it is omitted then all parameters will be treated as strings as they were in versions of Report Engine prior to 7.3/1071. NB agr.UseDataSetOnly also forces this option.

```
agr.UseParameter ParameterName, DefaultValue [, Type]
```

tells the system to use the specified parameter. Only parameters used in this way will be imported from Unit4 Business World. It is useful to specify a default value so that the report can also be run independently of Unit4 Business World during the development process. If specified the Type must be one of "integer", "string", "date", "float". If the given default value or the value passed in to the report is not of this type and cannot be converted to this type then an error occurs. If Type is not specified then the type is inferred from the type of the DefaultValue argument.

```
myvariable = agr.Parameter("parameter_name")
```

Gets the value of a parameter previously defined with agr.UseParameter.

When entering SQL in the **Edit | Query…** screen you use the prefix **$?** to get the parameter value:

```
… AND field = '$?parameter_name' …
```

Note Parameters can also be bound directly to Field Controls.

```
Sub OnDataInitialize

 ' parameters

 agr.UseTypedParameters

 agr.UseParameter "apar_type", "R"

 agr.UseParameter "my_date", "2005-02-22", "date"

 agr.UseParameter "one_point_three", 1.3

 …
```

6.2.1.6. Merge

```
agr.Merge("formula_name")
```

Where *formula_name* is a formula bound to a Rich Edit Control.

Causes all fields, formulas, titles, parameters etc within the text of the Rich Edit control to be replaced by their current values.

E.g. if the Rich Edit control contains the text:

This is some example text containing [data_field1] and some formatting.

Then the string [data_field1] will be replaced by the current value of the field named "data_field1" and all the formatting of the text will be preserved.

Please see sample "Merge.rpx"

Example:

```
Sub OnDataInitialize

 agr.Formula("Text1") = ""

 agr.Formula("Text2") = ""

 …

End Sub

Sub OnFetchData(eof)

 If eof Then Exit Sub

 If agr.IsGroupStart("resource_id") Then

       agr.Merge "Text1"

 End If

 …

 if agr.isgroupend("resource_id") then

       agr.Merge "Text2"

 end if

End Sub
```

6.2.1.7. Formula

```
myVariable = agr.Formula(formula_name)
```

Gets the value of a formula field named *formula_name*

```
agr.Formula(formula_name) = expression
```

Sets the value of the formula field named *formula_name* to the value of *expression*

When entering SQL in the **Edit | Query…** screen you use the prefix **$@** to get the formula value:

```
… AND field = '$@formula_name' …
```

Note Valid only in OnDataInitialize or OnFetchData.

When used to set a formula in OnDataInitialize then a new formula field will be created if it does not already exist.

Example:

This example uses a variable called **WholeAddr:**

```
agr.Formula("Address_40") = WholeAddr
```

This example uses an expression:

```
agr.Formula("CustAddr") = agr.field("faddr_apar")+ agr.field("faddr2_apar")
```

6.2.1.8. Field

```
myvariable = agr.Field("myfield")
```

Gets the value of the field named "myfield" from the report's query.

Note Only valid in Report_FetchData()

Example:

This example uses the variable **addr1**:

```
Addr1 = Mid (agr.Field("Address"), 1, 40)
```

6.2.1.9. IsGroupStart / IsGroupEnd

```
If agr.IsGroupStart(field) Then
…code goes here…
End If

If agr.IsGroupEnd(field) Then
…code goes here…
End If
```

Note Only Valid in OnFetchData().

Returns true if the current row is the Start or End of the break point/ group. **Field** should be the name of the field specified in the DataField property of a GroupHeader section in the report. If **Field** is a zero length string then these properties can be used to identify the first or last row in the recordset.

Example:

This example sets the value of variable **Tot_NotDue** to zero:

```
If agr.IsGroupStart("currency") Then

 agr.Formula("Tot_NotDue") = 0

End If
```

6.2.1.10. UseParentControl / ParentControl

```
agr.UseParentControl parent_control_name, default_value
```

declares that the sub-report references the specified Textbox Controlin the parent report. It is useful to specify a default value so that the report can also be run independently of Unit4 Business World and independently of the parent report during the development process.

```
sSql = sSql & "…" & agr.ParentControl(parent_control_name)
```

Gets the DataValue property of the Textbox Control in the parent report.

Note Valid only in a sub-report.

```
agr.ParentControl(parent_control_name) = myDataValue
```

Sets the DataValue property of the Textbox Control in the parent report.

Note Valid only in a sub-report.

When entering SQL in the **Edit | Query…** screen you use the prefix **$!** to get the parent control value:

```
…AND field = '$!parent_control_name'…
```

In order to use a Textbox Control from the parent report in a sub-report it must first be declared using agr.UseParentControl.

Examples

```
agr.UseParentControl "txtvoucherno", 39980001
```

As used in **Edit | Query…**

```
…AND T.voucher_no = '$!txtvoucherno'…
```

As seen in vbScript

```
…where t.voucher_no = agr.ParentControl("txtvoucherno")…
```

6.2.1.11. ParentTag

```
sTag = agr.ParentTag
```

Returns the value of the "Tag" property of the sub-report control containing this instance of the sub-report. If the report is not a sub-report then a zero length string will be returned.

This property can be very useful if you wish to have two instances of the same sub-report within the same section of the main report. E.g. you might wish to run two instances of the report one of which will be marked as a copy with code like this:

```
if agr.ParentTag = "copy" then
 rpt.lblCopy.Caption = "COPY"
else
 rpt.lblCopy.Caption = ""
end if
```

6.2.1.12. SetDefaultCanvas / SetCanvas

```
agr.SetDefaultCanvas canvas_name
```

Defines the Canvas that will be used to underlay each report page unless overridden using agr.SetCanvas.

Note Valid only in OnDataInitialize()

```
agr.SetCanvas page_number, canvas_name, [canvas_type]
```

Defines the Canvas that will be used on the specified page.

By default the Canvas is used to underlay the page (**canvas_type** = 0).

Set **canvas_type** to 1 if you wish to insert the Canvas before the specified page.

Set **canvas_type** to 2 if you wish to insert the Canvas after the specified page.

Example

```
Sub OnBeforePrint
 select case rpt.txtClient
 case "EN"
       agr.SetCanvas rpt.txtPageNumber, "SetCanvas1"
 case "NO"
       agr.SetCanvas rpt.txtPageNumber, "SetCanvas2"
 end select
End Sub
```

Sample

SetCanvas.rpx

6.2.1.13. SetPaperBin / SetDefaultPaperBin

```
agr.SetDefaultPaperBin bin
```

Sets the paper bin that will be used for each report page unless overridden using agr.SetPaperBin. Only valid in OnDataInitialize().

```
agr.SetPaperBin page_number, bin
```

Sets the Paper Bin that will be used on the specified page.

Example

```
Sub OnBeforePrint
 select case rpt.txtClient
 case "EN"
       agr.SetCanvas rpt.txtPageNumber, 1
 case "NO"
       agr.SetCanvas rpt.txtPageNumber, 2
 end select
End Sub
```

NOTE: Paper bins are printer specific. Some printers require bin numbers like 1 or 2 corresponding to the number embossed on the physical device. Others require numbers like 257 and 258.

Sample

SetPaperBin.rpx

6.2.1.14. SetQuery

```
agr.SetQuery sql
```

sets the sql select statement to be used as the main driving query for the report.

agr.SetQuery **AGRESSO** means take the data given by the output of an arw report or Browser screen.

agr.SetQuery **AGRESSO** *template_name* means run the specified browser template and take the data from that

agr.SetQuery **TABLE *my_table_name*** means take the data from the specified table name within an externally supplied data set.

NOTE: It is always possible and better (more efficient) to write the sql within the report than to use the data output from an arw report.

Valid only in OnDataInitialize.

Examples:

This example uses an ARW:

```
'#Query - this code is automatically inserted by 'Edit>SQL'.  If you change it
here you will no longer be able to use the SQL editor

 agr.SetQuery "AGRESSO"

'#End Query 10627
```

This example uses code that has been entered via the **Edit | Query…** screen:

```
'#Query - this code is automatically inserted by 'Edit>SQL'.  If you change it
here you will no longer be able to use the SQL editor

 Dim sQuery

 sQuery = "SELECT client, client_name"

 sQuery = sQuery & vbCrLf & "FROM acrclient"

 sQuery = sQuery & vbCrLf & "ORDER BY client"

 agr.SetQuery sQuery

'#End Query 81111
```

6.2.1.15. SelectInto

```
agr.SelectInto sql [, prefix]

or

nRows = agr.SelectInto(sql [, prefix])
```

Note sql means a SQL statement

Runs *sql* and sets the results into a formula for each of the returned columns. The Select statement should return zero or one row. If no row is returned then the variables are set to appropriate defaults for their data type.

If specified *nRows* is set to the number of rows returned by the SQL statement (0 or 1)

If "prefix" is specified the this will be added to the front of the returned column names to give the formula names.

Valid only in OnDataInitialize and OnFetchData events.

Note Using this method can very often simplify the main Select statement of the report and remove the need to handle outer joins.

Note If used in the OnFetchData event it will be necessary first to create the formulas within the OnDataInitialize() event using agr.Formula("column_name") = initial_value

Example

```
agr.SelectInto("SELECT rel_value as Del_Type" & _

  " FROM aglrelvalue " & _

  " WHERE att_value = '" & agr.Field("apar_id") & "'" & _

  "   AND attribute_id = 'A4' " & _

  "   AND client = '" & agr.Parameter("client") & "'" & _

  "   AND rel_attr_id = 'D3' ")
```

6.2.1.16. GetRecordSet

```
set rs = agr.GetRecordSet(sql)
```

Runs *sql* and returns a corresponding ADO recordset.

Sample

GetRecordSet.rpx

6.2.1.17. GetFile

```
MyFile = agr.GetFile(fileName)
```

Reads the entire contents of the specified file from the file system. This is often useful for including a company logo in a report. It may reduce the size of the rpx file considerably.

Note. If filename is not relative then it is taken to be relative to the location of the rpx file that is currently executed. Often the "Customized Reports" folder.

Example

```
' get the company logo

agr.Formula("logo") = agr.GetFile("my_logo_file.jpg")



(the formula "logo" would then be used as the data source for a picture
control)
```

6.2.2. **Unit4 Business World (former Agresso) Specific Methods**

6.2.2.1. GetSysConf

```
myValue = agr.GetSysConf([name], [client])
```

Returns an Unit4 Business World system configuration code

Example

In the OnDataInitialize Event of the Report you might wish to find the system parameter for the relation attribute id to display on the report:

```
agr.Formula("rel_attr_id1") = agr.GetSysConf("UKP_REL_ATTR_ID1")
```

6.2.2.2. GetText

```
mytext = agr.GetText(TextType, Variant, Language)
```

Returns the requested Unit4 Business World text (used for Statement text; Reminder text, etc.)

6.2.2.3. GetAddress

```
myaddress = agr.GetAddress(AttributeId, DimValue, [Client], [AddressType],
[SequenceNo])
```

Returns the address for the given attribute value in the format defined in Unit4 Business World screen AG40 (Address setup).

Example

```
agr.Formula("Address") = agr.GetAddress("A5", agr.field("apar_id"))
```

6.2.2.4. GetDescription

```
mydescription = agr.GetDescription(AttributeId, AttributeValue, [client],
[language])
```

Returns the description of the *`AttributeValue`* assuming that it is a valid value of attribute specified by *`AttributeId`*. Client and language are taken from the corresponding parameters if not supplied.

Example

```
If agr.IsGroupStart("invoice_ref") Then

 agr.Formula("Deliv_Descr") = agr.GetDescription("D3", agr.formula("Del_Type"))

End If
```

6.2.2.5. GetDocument

```
MyDocument = agr.GetDocument(client, DocType, Key1, [Key2,] … [Key6])
```

Gets the specified document from the Unit4 Business World document archive.

Example

```
  ' get the photo

' arguments are:

'client

'DocType

'Key1

'Key2

'...

'Key6

agr.Formula("photo") = agr.GetDocument(agr.Parameter("client"),
msDocType,agr.Parameter("client"), agr.Field("resource_id"))
```

6.2.2.6. UseTitle / Title

```
agr.UseTitle title_name, default_value
```

Tells the system to use the specified title. Only titles used in this way will be imported from Unit4 Business World. It is useful to specify a default value so that the report can also be run independently of Unit4 Business World during the development process.

```
mytitle = agr.Title(title_name)
```

Returns the Unit4 Business World title in the current language.

Valid only in OnDataInitialize and OnFetchData

Note Titles can also be bound directly to Field Controls.

Example:

```
    agr.UseTitle "interest", "Int. rate"
```

6.2.2.7. UseTitles

```
    agr.UseTitles table_name
```

When working against a DataSet (agr.ParserType = 3) allows titles to be loaded from the specified table in the dataset. The table should contain exactly one row

AttachDocument

```
    agr.AttachDocument client, DocType, Key1, [Key2,] … [Key6]
```

Specifies that the Named document should be attached to the email when the report is distributed. This is a better alternative to using the xa_ parameters. This could be used to attach a document for each invoice line for example.

The arguments are as per agr.GetDocument

6.2.3. **Utility Methods**

6.2.3.1. ToDate

```
mydate = agr.ToDate(date_string)
```

converts *date_string* to a date value. Has the same effect as CDate() except that formats like **yyyymmdd**, **yyyymmdd hh:mm** or **yyyymmdd hh:mm:ss** (as for date type parameters passed from Unit4 Business World) are also acceptable and that if *date_string* is a zero length string then 1900-01-01 00:00:00 is returned.

Example:

In the OnDataInitalize Event of the Report you might use the following code:

```
agr.Formula("date_from") = agr.ToDate(agr.Parameter("date_from"))
agr.Formula("date_to") = agr.ToDate(agr.Parameter("date_to"))
```

In the OnFetchData Event of the Report you might then use the following code:

```
If agr.Field("date_from") <= agr.Formula("date_to") And _
   agr.Field("date_to") >= agr.Formula("date_from") Then
 agr.Formula("back_colour") = vbRed
Else
 agr.Formula("back_colour") = vbYellow
End If
```

You can then use the "back_colour" formula to highlight those rows that overlap the given dates.

6.2.3.2. FormatSQLDate

Please note that agr.FormatSQLDate() is superseded by agr.QuoteSQL()

```
mysql = mysql & " AND " & agr.FormatSQLDate(mydate) & "BETWEEN…"
```

Given a date argument this returns a string suitable for inclusion in an SQL statement.

Example:

In the OnFetchData Event of the Report you might wish to find resource relations for the payment date:

```
sSql = "SELECT r.rel_value" & _
       "  FROM ahsrelvalue r" & _
       " WHERE r.client = '" & agr.Parameter("client") & "'" & _
       "   AND " & agr.FormatSQLDate(agr.Formula("pay_date")) & _
       "       BETWEEN r.date_from AND r.date_to" & _
       " AND r.rel_attr_id = '" & msRelAttrId & "'" & _
       " AND r.resource_id = '" & agr.Field("resource_id") & "'"
agr.SelectInto(sSql)
```

6.2.3.3. QuoteSQL

```
mysql = mysql & " AND " & agr.QuoteSQL(mydate) & "BETWEEN…"
```

or very often:

```
mysql = mysql & " AND " & agr.QuoteSQL(agr.Parameter("my_date_parameter")) &
"BETWEEN…"
```

Quotes a variable in a format suitable for inclusion in an SQL statement.

E.g. if *value* is a string equal to *abc* it will return *'abc'.* If *value* is a date it will return *$date('yyyy-mm-dd hh:nn:ss')*, if *value* is an number equal to *12.34* it will return *12.34* (even if the decimal separator is a comma in the current locale)

agr.QuoteSQL("my string") will return 'my string'

agr.QuoteSQL(myDateVariable) will return $date('yyyy-mm-dd hh:nn:ss')

agr.QuoteSQL(12.34) will return 12.34 (with a "." whatever the current regional settings)

Example:

In the OnFetchData Event of the Report you might wish to find resource relations for the payment date:

```
sSql = "SELECT r.rel_value" & _
  " FROM ahsrelvalue r" & _
  " WHERE r.client =agr.QuoteSQL(agr.Parameter("client")) & _
  " AND " & agr.QuoteSQL(agr.Formula("pay_date")) & _
  " BETWEEN r.date_from AND r.date_to" & _
  " AND r.rel_attr_id='" & msRelAttrId & "'" & _
  " AND r.resource_id=" & agr.QuoteSQL(agr.Field("resource_id"))

agr.SelectInto(sSql)
```

6.2.3.4. Format

```
myvariable = agr.Format(expression, format)
```

Converts expression to a string usually for use in a formula.

In fact this simplifies the VB `Format` function as it offers better functionality than the VBScript `FormatDate(), FormatNumber()` etc.

6.2.3.5. Log

```
agr.Log message
```

Writes *`message`* into the log file.

Useful for debugging.

Example

```
agr.Log  (sTmpMonth)
```

6.3. EVENTS

The following events are available from VB Script:

6.3.1. Report Events

6.3.1.1. OnDataInitialize()

This event is fired once when the report starts to run and is the only point at which new data fields can be created.

All calls to **agr.UseParameter**, **agr.UseTitle**, **agr.UseParentControl** and the initialization of all **Formulas** must appear here (or in a subroutine called from here).

It must also contain an **agr.SetQuery** statement to define where the data is to come from and create the **Fields**.

6.3.1.2. OnFetchData(eof)

This event is called sequentially for each row of data with eof (end of file) set to **false** and then one more time with eof set to **true**. At the point where the extra call with eof set to **true** is made the Fields are still set to their values for the last row.

It is here that all data processing should occur and all formulas be evaluated because this is the only event that is guaranteed to be fired in sequence once for each row. The **agr.IsGroupStart()** and **agr.IsGroupEnd()** are essential to identify rows that will be bound to the corresponding Group Header and Group Footer sections.

In this section you **must not** refer directly to any controls in any section of the report.

The **only** way of passing data from here to the controls is to set up a formula and bind a control to it.

6.3.1.3. OnReportStart()

This event fires after **OnDataInitialize** but before the first **OnFetchData**.

It is useful for adjusting the report, perhaps to change the DataField property of Controls.

6.3.2. **Section Events**

The report consists of the following sections:

- Report Header
- Page Header
- Group Headers
- Detail
- Group Footers
- Page Footer
- Report Footer

You cannot guarantee the order in which these are processed.

There are three events that occur for every section:

- OnFormat
- OnBeforePrint
- OnAfterPrint.

Within each event, you must only refer to controls that occur on the report surface in the current section.

You can add a non-visible control bound to a formula onto Group Header section, if you wished to refer to this within the format event. For example, if you wished to change which header wording you used, dependent upon a field in the dataset, you would need to add a non-visible control to the Group Header, which was bound to the client data field. Then you could refer to this within the OnFormat event for the Group Header section.

In these events you can refer to controls **within this section only**. It is invalid to refer to controls outside this section or data fields, i.e. agr.Field(), agr.Formula(), agr.Title() and agr.IsGroupStart() cannot be used. If you want to use the value of a data field then you should include a hidden control bound to the data field.

6.3.2.1. OnFormat()

This is fired before the data is laid on the page and can be used to modify the layout of the section or any control within the section. The format of the section and the controls it contains can be changed including changing its height or making it invisible.

Example

If you wished to change which company logo you used, dependent upon the client you have run the report from, you would need to add a non-visible control to the Group Header, which was bound to the client data field. Then you could refer to this within the OnFormat event for the Group Header section.

Note

You **cannot** use this event to process data, evaluate accumulators etc. This must be done in **Report_OnFetchData()**

6.3.2.2. OnBeforePrint()

This event fires when the section is ready to be printed.

At this point you can access the values of summary fields (including page numbers and total page counts)

Within this event you can modify the control values, but you cannot alter properties such as Height.

Example

If you use the menu item **Page N/M** from the Insert menu, the coding needed will be added to this event for the section automatically.

6.3.2.3. OnAfterPrint()

This is the final event that fires after the section has been printed.

At this point it is too late to change anything. However, this event is where you should add entries to the Table of Contents (TOC).

6.4. PORTABLE SQL FUNCTIONS

These functions are now deprecated and should not be used in new reports. Please use ASQL syntax instead which is now always available.

A number of special portable SQL functions are available which will produce the correct results on any supported DBMS or connection mechanism. They all have the form

```
$function([expression], [expression], …)
```

The functions take zero or more expressions as arguments but the brackets are mandatory even for functions with no arguments. The following functions are currently supported:

6.4.1. $TO_CHAR

```
$to_char(expression)
```

Converts `expression` to a character value, e.g.

```
SELECT * FROM acuheader WHERE $to_char(apar_id) = '123'
```

6.4.2. $DATE

```
$date(date_literal)
```

Converts *date_literal* to a date value. *date_literal* must be in one of the following formats:

```
'yyyymmdd hh:nn:ss'

'yyyymmdd hh:nn'

'yyyymmdd'

'yyyy-mm-dd hh:nn:ss'

'yyyy-mm-dd'
```

```
SELECT * FROM acutrans WHERE due_date < $date('20040131 00:00:00')
```

NB the format used for date parameters in Unit4 Business World server processes is yyyymmdd hh:nn:ss

6.4.3. $IFNULL

```
    $ifnull(expression, value)
```

Returns *value* if *expression* is null otherwise returns *expression*.

(Equivalent to IsNull() in SQL Server or NVL() in Oracle)

6.4.4. $NOW

```
$now()
```

Evaluates to the current date/time, e.g.

```
SELECT $now(), * FROM acrclient
```

NB the brackets are needed.

6.4.5. $MID

```
    $mid(expression, n, m)
```

Returns a substring of `expression` starting at character n and m characters long, e.g.

```
    $mid('abcdef',3,2)
```

will return 'cd'

6.4.6. $LEFT

```
    $left(expression, n)
```

Returns a substring of `expression` starting at character 1 and n characters long, e.g.

```
    $left('abcdef', 2)
```

will return 'ab'

6.4.7. $RIGHT

```
$right(expression, n)
```

Returns a substring consisting of the rightmost n characters of *`expression`*, e.g.

```
$right('abcdef', 2)
```

will return 'ef'

6.4.8. $CONCAT

```
    $concat(expression1, expression2 [,expression3 [,expression4,…]])
```

Returns the concatenation of *expression1* and *expression2*, e.g.

```
    $concat('abcdef','gh')
```

will return 'abcdefgh'

up to 8 expressions may be concatenated.

6.4.9. $LPAD

```
$lpad(expression, n, ['c'])
```

Returns a string consisting of *`expression, n`* characters long padded on the left with character 'c', If not specified the string is padded with spaces e.g.

```
$lpad('123',5,'0')
```

will return '00123'. This is useful for performing a numeric sort on numbers stored as strings.

6.4.10. $MOD

```
$mod(expression1, expression2)
```

Returns the value of *`expression1`* modulo *`expression2`*, e.g.

```
$mod(3,2)
```

will return 1

6.4.11. Zero Length String

```
' '
```

Not exactly a function but '' (a literal zero length string) will be converted to ' ' (a literal single space) when running on an ORACLE database. This is because Unit4 Business World uses a single space to represent a zero length string when running on ORACLE. So that

```
SELECT count(*) FROM agldescription WHERE dim_value = ''
```

will have the same effect on ORACLE as it does on MS SQL Server.

6.5. REPORT DESIGNER

When you open the AGRESSO Report Creator Designer, you will see three tabs:

6.5.1. Design Tab

This page is part of the standard layout for the Agresso Report Writer.

6.5.2. Canvas Tab

The canvas, in its simplest form, contains all of the information you would have traditionally placed on to pre-printed stationery.

You can draw lines and boxes on the report itself, if they are restricted to the header and footer sections, or on the canvas, if they are to continue to the bottom of pages.

Ensure that your canvas has the same page and print width settings as your main report.

Use the double-headed arrows at the edge of the page to make the white area of the canvas big enough to hold the box that you are going to draw.

Select **Canvas | Save** and save this with the same name as the report itself – you will notice that the canvas will have a **.cnv** extension.

As long as the files are both stored in the same folder (e.g. the Customised Reports folder), they will continue to work together.

6.5.3. Preview Tab

This provides a preview of the report output.

6.5.4. **Menus**

6.5.4.1. File Menu

New	Open a new report
Open…	Open an existing report
Save	Save the open report
Save As…	Save the open report under a different name
Export…	This allows you to save the report as a range of differing files, e.g. as a pdf file. Note this option is only available in the Preview tab
Print setup…	Accesses the standard Windows print setup screen, where page size and orientation can be set
Page setup…	There are buttons here enabling you to alter the page setup, the printer setup, the grid settings and the style properties
Exit	Closes the AGRESSO Report Creator

6.5.4.2. View Menu

Grid	Show or hide the grid on the reporting page
Explorer	Show or hide the Explorer Toolbar
Log file…	View the log file from the most recent previewing of the report

6.5.4.3. Canvas Menu

These options all concern the Canvas tab:

New	Open a new canvas
Open…	Open an existing canvas
Save	Save the open canvas
Save as…	Save the open canvas under a different name

Close Close the open canvas

6.5.4.4. Edit Menu

Find… This will find all the instances of a string in the report, i.e. not just in the script but also in the report. This is useful to find, for example, where a field is used as it will point you to the Controls it is linked with as well as the script sections that contain it

Query… Edit the query that creates the dataset for the report.

Script… Edit the script that runs behind the report.

Datafield… Alter the datafield that a control is bound to

Caption… On a label control, edit the caption that you originally entered

Report Name… Allows you to change the subreport that is linked to your report. Click on the subreport before choosing the option.

6.5.4.5. Insert Menu

Page number… Inserts the page number for the page – this can be part of the numbering for the whole report or for a group within the report

Total page count… Inserts the total page number – this can be part of the numbering for the whole report or for a group within the report

Page N / M… A special function that adds in the page number and the total page count, preceded by the word page. This adds hidden controls to the report and code into the relevant section of the script

Date… Inserts the current date and time – this can be formatted to show just the date or just the time

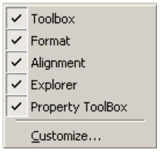
6.5.4.6. Tools Menu

Review control names	If you have altered the data fields that a textbox is bound to, you can simply choose this option to re-set the names of the controls. On occasions this may also be useful at other times too, for example when the report appears to have lost the connection with the data.
Map data fields	This looks through the report allowing user to map any references to non-existent data fields

6.5.4.7. Help Menu

Help	Access the Unit4 Business World Report Engine helpfiles
About AGRESSO Report Creator…	Find the build and version number of the software

6.5.5. Toolbars and Toolboxes



- ✓ Toolbox
- ✓ Format
- ✓ Alignment
- ✓ Explorer
- ✓ Property ToolBox

Customize...

You can close and delete the toolbars and boxes – sometimes without meaning to.

Click on the grey area at the top of the screen with your right mouse button and you can choose to display them again.

6.5.5.1. Design Toolbox

Pointer changes the cursor back to a pointer

Label adds a label control to the report

Textbox adds a textbox control to the report that can be linked to fields, parameters, formulas, group start/end or titles

Checkbox adds a checkbox control to the report

Image adds a control to the report that can be linked to an image via the Picture property

Line adds a line to the report

Shape adds a shape to the report – the choice is from a rectangle, ellipse or rounded rectangle. The choice is controlled via the Shape property, with rectangle being the default

RichEdit Control adds a rich edit control to the report

Frame adds a frame to the report

Subreport Control helps to add a subreport to the report

PageBreak a method of adding a page break to the report. Perhaps a more useful method is via the NewPage property on a group header or footer

OLE Object is for advanced use only and is not supported by Unit4 Business World

ActiveX Control only to be used for charts using MSChart. Other items here use code which is external to the AGRESSO Report Creator and as such are not supported by Unit4 Business World

Barcode Control allows insertion of a bar code that can be linked to fields, parameters, formulas, group start/end or titles

ADO Data Control you cannot use Data Controls with Agresso Report Creator

DAO Data Control you cannot use Data Controls with Agresso Report Creator

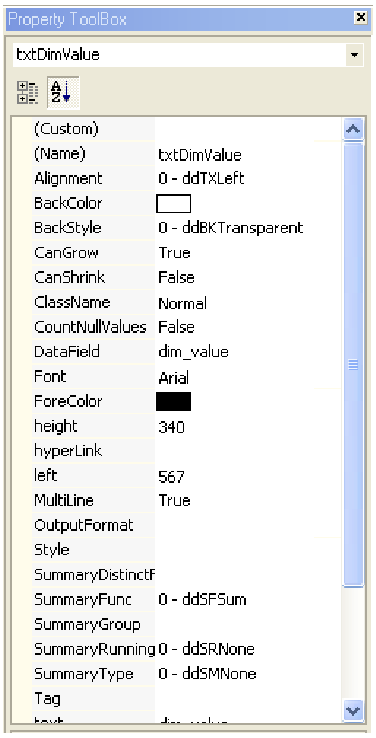
RDO Data Control you cannot use Data Controls with Agresso Report Creator

XML Data Control you cannot use Data Controls with Agresso Report Creator

6.5.5.2. Property Toolbox

In addition to the formatting toolbars, there is also an object property toolbox on the right. The contents of this toolbox will change, depending on what you have highlighted.

You can also use the drop-down box at the top to choose a different object.



txtDimValue

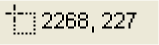
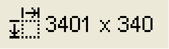
(Custom)	
(Name)	txtDimValue
Alignment	0 - ddTXLeft
BackColor	□
BackStyle	0 - ddBKTransparent
CanGrow	True
CanShrink	False
ClassName	Normal
CountNullValues	False
DataField	dim_value
Font	Arial
ForeColor	■
height	340
hyperLink	
left	567
MultiLine	True
OutputFormat	
Style	
SummaryDistinctF	
SummaryFunc	0 - ddSFSum
SummaryGroup	
SummaryRunning	0 - ddSRNone
SummaryType	0 - ddSMNone
Tag	
text	dim_value

6.5.5.3. The Format Toolbar

Normal ▾	Select a style for the paragraph
Arial ▾	Select and applies a font to selected text
10 ▾	Select a point size for a font
B	Makes selected text bold
I	Makes selected text italic
U	Adds an underline to selected text
≡	Aligns selected text to the left of the paragraph
≡	Aligns selected text in the centre of the paragraph
≡	Aligns selected text to the right of the paragraph
☰	Applies bullets to a paragraph
⇤	Decreases an indent
⇥	Increases an indent
▾	Displays the fill colour palette to fill an object
A ▾	Displays the fill colour palette to change font colour
✎ ▾	Displays the fill colour palette to change the line colour
☰ ▾	Displays the line palette to select a line type

6.5.5.4. The Alignment Toolbar

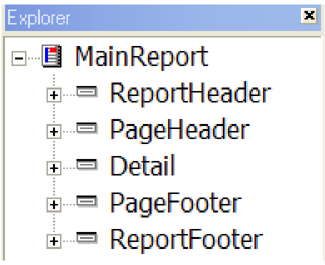
⧉	Bring highlighted control(s) to front
⧉	Send highlighted control(s) to back
🔒	Lock controls
⌗	Switch on/off snap to grid – when this is switched on, any new controls will automatically line-up to the grid
⌗ ▾	Align controls

	Make controls the same width
	Make controls the same height
	Make controls the same size
 2268, 227	Position of the control
 3401 x 340	Size of the control

6.5.6. Report Explorer

Every report is broken up into sections. This pane on the left-hand side of the screen shows the current sections of the report.

If you click on the plus signs, you can also see all of the controls that are on the report, in their relevant sections.



Explorer
- MainReport
  - ReportHeader
  - PageHeader
  - Detail
  - PageFooter
  - ReportFooter

You can also click on an item in this pane and it will be automatically shown on the report.

- The **Report Header** section may contain introductory information
- The **Page Header** section will often contain the titles of columns
- Any **Group Headers** will go next, in order
- The **Detail** area will contain textbox controls bound to data fields
- The **Page Footer** may contain the page numbering options or running totals
- Any **Group Footers** will follow, in reverse order
- The **Report Footer** is a useful place for summing up the statistics such as Grand Totals etc.

6.6. TECHNICAL NOTES

AGRESSO Report Creator is built around ActiveReports 2.0 from Data Dynamics (now “ComponentOne, a division of GrapeCity”).

AGRESSO Report Creator uses VBScript. For a complete language reference for VBScript please refer to http://msdn.microsoft.com/scripting. However, please note that not all the VBScript functionality is applicable to AGRESSO Report Creator.

Note

There are various sample reports that have been designed to help demonstrate concepts. Where appropriate, these are indicated at the bottom of topics.

6.6.1. Multiple Canvases

It is possible to have different canvases for different pages in the report.

To define a canvas for the whole report use SetDefaultCanvas

This only needs to be set if your default canvas has a different name from the report.

The default can be overridden on a page-by-page basis using the SetCanvas method.

Sample

SetCanvas.rpx

6.6.2. The `agr` and `rpt` Objects

6.6.2.1. Script…

The reports are all written in **VB Script**. Through the VB Script we have access to **objects**. Objects are a little like links to libraries of code that can be used. So using an object will provide access to various collections, properties and methods.

There are two main objects that you would be using within AGRESSO Report Creator:

6.6.2.2. The `agr` Object

From the agr object, there are three main collections of values:

- **Field** links to a field that is contained within the dataset
- **Formula** is created and initialised within the script
- **Parameter** passes information from the report ordering screen (in a server environment) and is initialised in the script

Example

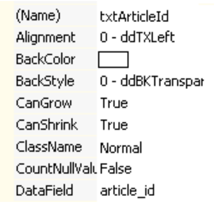
```
agr.Formula("inc VAT") = agr.field ("amount") *1.175
```

The formulas and parameters all need to be initialised at the top of the script, before they can be used. When you create a formula in AGRESSO Report Creator, the software sets this up for you.

6.6.2.3. The `rpt` Object

This object allows you to reference the sections and controls on the report.

Example



(Name)	txtArticleId
Alignment	0 - ddTXLeft
BackColor	□
BackStyle	0 - ddBKTranspar
CanGrow	True
CanShrink	True
ClassName	Normal
CountNullValu	False
DataField	article_id

The name of a textbox control here can be used as shown below:

```
If rpt.txtarticleid=0 then
 rpt.txtClient.visible = false
Else
 rpt.txtClient.visible = true
End if
```

6.6.3. **Multiple SELECT Statements**

6.6.3.1. Writing the SQL

SetQuery creates the main dataset for the report. You can use the keyword AGRESSO to access a dataset that has been created by an ARW, or write the SQL yourself.

You can access the query through Edit | Query or via the Script window (**Note** you would generally endeavour to alter this through the query window and allow AGRESSO Report Creator to update the code in the Script window).

This is the main Select statement.

With **SelectInto** you write the SQL to return a single row of data, rather than a dataset. The data is returned into the formulas so that you can access it.

Sample

GL09.rpx

6.6.3.2. Advanced Feature

With **GetRecordSet** you write the SQL and you need to handle the data that it returns. It returns an ADO recordset that you then process in the VBScript.

Sample

GetRecordSet.rpx

6.6.4. Multi-lingual Texts

The multilingual texts are looked up from Unit4 Business World for the language defined by the parameter "language". This can be overridden with the parameters "xw_language" and "xs_language". The titles available for a standard report are held in **asysrepref**. On top of this you can define numeric titles and these will be looked up in the appropriate **asystitles** table.

Example

Report GL11 has the following title in **asysrepref**: **t_client**. If you define it in the RPX file using `agr.UseTitle "client", "Client"` then it will substitute in the value for the correct language when you run the report.

If you have a user-defined report in which you want to use the language independent version of **client** you use `agr.UseTitle "50257", "Client"`.

Where **50257** is the number of the title from **asystitles** that you wish to use in your report.

AGRESSO Report Creator will substitute the correct value from the correct version of **asystitles** when you run the report.

6.6.5. **Golden Rules**

6.6.5.1. Do

14. USE Bound Fields. I.e. set the DataField property of all Field Controls to the name of a field (from the recordset), a formula (defined in OnFetchData), a parameter or a title.
15. DO ALL data processing, calculations, subtotalling etc in OnDataInitialize / OnFetchData – make use of agr.IsGroupStart() and agr.IsGroupEnd(). NEVER use Section_OnFormat() events for this purpose
16. USE Tools | Verify – it is there to enforce the rules and prevent you from producing incorrect reports. (It is quite possible to write reports incorrectly which look right some of the time).
17. USE the Insert Text box button or Edit | Sql to generate VBScript for you. It is much easier than starting with an empty screen.

6.6.5.2. Do not

18. DO NOT use Page Headers to display information which changes during the report. USE a Group Header with the "Repeat" property set to **4 Every Page including no detail**.
19. NEVER refer to the fields collection outside of OnDataInitialize or OnFetchData. This is because AGRESSO Report Creator "fetches ahead" so that outside of these events you can never assume that the fields collection with contain the values from the correct row.
20. NEVER rely on the order that the OnFormat events are fired in. Sections are formatted in advance in order to determine if they will fit on the same page etc.
21. NEVER refer to a Field Control outside its own section.
22. DO NOT refer directly to the rpt.fields collection – Use agr.field, agr.title, agr.parameter, agr.formula they provide much better error handling and enforce some of the other rules
23. DO NOT attempt to implement global variables using agr.formula() in order to communicate between the Section_OnFormat() events because you cannot rely on the order in which these events fire.
24. There are no intrinsic global variables.
25. The Section_OnFormat() events should only be used for adjusting the format of the current section – never for data processing. If you need to refer to values other than those of the visible fields in the section then include hidden controls bound to the data that you want to use
26. Putting VBScript into a DataField property (preceeded by an "=") is almost always the wrong thing to do because you can only refer to other field controls in the same section not items in the fields collection.
27. DO NOT use page numbering where the page break is in a deeper group than the group you are putting the page number into, as the results may be unpredictable.

7. CSV REPORTER

This is a very simple Report Renderer that simply creates a csv file from the first (or a named) table within the dataset. There is not input file, the process is controlled by the following parameters

Parameter	Purpose	Default value
csv_table	defines which table the csv file will be created from	The first table
csv_list_separator	Defines the list separator to be used	comma ‘,’ if English otherwise semicolon ‘;’
csv_decimal_separator	Defines the decimal separator to be used	Full stop ‘.’ if English otherwise comma ‘,’
csv_quote_character	Defines quote character for strings	Double quote “
csv_header_row	0/1 indicates that header row (column names) should be included	1 = true
csv_quote_all_strings	set to indicate that all strings (including those not themselves containing the quote character) should be quoted	0 = false
csv_datetime_format	sets the format to be used for dateTime columns	yyyy-MM-dd HH:mm:ss
csv_nobom	0/1 suppresses the unicode "byte order mark" (3 special bytes at the start of the file)	0 = false (Byte order mark present)
csv_final_new_line	0/1 indicates that a final new line character should be included after the last data row	1 = true

Please see the sample report named “Create CSV File.rerx” for a heavily annotated example

8. REPORT PACKAGE FILES [.RERX]

Report package files are a way of joining all report files together with the data required to run/render the report and produce the output.   They are opened by the Report Engine client application and consequently Report Engine must be installed on the client machine in order to process these files.

Report package files contain all or some of the following constituent files:

8.1. DATA EXTRACTION (.RESQL) FILE

This file contains instructions to drive the Data Extraction process that will create the .NET DataSet.

8.2. _DATA.XML

This is simply a copy of the .NET DataSet containing the data for the report persisted as in xml format.

8.3. REPORT FILE(S)

The input report definition file(s) e.g. myreport.xlsx or myreport.rpx

8.4. PROPERTIES

Properties are held in a hidden file called _properties.xml. The properties that control the way that the report package will be rendered.

8.4.1. FileName

This is the name of the primary report file. If the report package contains only one report this is not mandatory, however, it is needed in the case of an ARC report that contains subreport .rpx files as well as the main .rpx file.

FileName may also contain the special syntax "!new.*report_type*" where *report_type* is one of the legal input report types: ".xlsx", ".xls", ".docx", ".doc", ".rpx", ".xml".

8.4.2. Reporter

This allows the user to specify which reporter object to use to render the report. Normally, this is defaulted from the input file type.

8.4.3. Timeout

The rendering process will timeout after this amount of time.

8.4.4. OutputType

This is a used to override the default output file type for the given input file type. Valid output types are:

Report Type	Output Type
.xlsx, .xls (Excelerator)	.xlsx, .xls, .pdf, .mht
.rpx (ARC)	.rdf, .pdf, .rtf, .tif, .txt, .htm, .mht, .xls
.docx, .doc (Worderator)	.doc
.xml (OpenXML)	.xlsx, .docx, .xml, .csv

9. RERX REPORT SPLITTING

Report splitting occurs at the ".rerx"/ ReSQL level not at the individual report file (".repx", ".xlsx", ".rpx") level. The mechanism is exactly the same as it has always been when running legacy (non-.rerx) ARC reports.

Splitting is triggered by the presence of the "xs_breakcol" parameter and a ".query main" or ".splitQuery *tablename*" in the ReSQL. The "xs_breakcol" parameter may be "declared" in the ReSQL or passed in externally (from the command line or from the Business World report ordering screen).

The distribution of the split report output is driven by special parameters (xf_..., xe_..., xd_..., xp_...) EXACTLY as was the case with the legacy ARC reports.

The process of splitting the report is straight forward and runs as follows…

- First the .rerx is processed down to the end of .query main / .splitQuery in order to extract the query/table that is going to drive the split. (GetDrivingTable)
- The entire .rerx is re-processed once for the whole report and then once for each split.
- In the whole report the main table will contain all rows, in the split reports it will contain only the rows for the current value of the break column.
- In the split report the special parameter $breakcol is available and equal to the current value of the split column.

And that is all. There is no more complexity than that.

This has some implications for how reports should be written if they are intended to be split.

- ReSQL files should contain ".query main" or ".splitQuery" and this should be as close to the top of the file as possible.
- Use .Loop main to gather the right rows for the split reports
- Use .AddParentRow to add exactly one row to a parent table
- Use .AddChildRows to add one or more rows to a child table (often the driving table for a sub-report)
- $breakcol can be used to determine if you are a split instance of the report
- You may want to use .if / .else / .endif within an SQL statement according to the value of $breakcol
- Use xs_report to select a different report file from within the report package (.rerx file) if required. (xs_inrep is not appropriate for rerx reports).
- If you specify xs_allow_changes you may change the values of the xe_*, xd_* and xp_* parameters within the resql AFTER .splitQuery. This allows you to gather the information required individually for

each split rather than being forced to include these values in the driving table which can often be cumbersome and almost always inefficient.

10. SPECIAL PARAMETERS

The following Unit4 Business World parameters have special meaning to Report Engine.

10.1. REPORT SPLITTING (XS_, XW_)

ParamID	Param name	Example value	Comments
xs_breakcol	Break Column	apar_id apar_id,voucher_no	Name of column to break on or comma separated list of columns
xs_allow_changes	Allow changes	1	Set to allow xe_, xd_ and xp_ parameters to be changed in the resql after .splitQuery
xs_filetype	File Type of split files	pdf	Any of the supported output file types for the given report type.
xs_print	Print Split Report File	1	Determines if split should be printed as they are created. If set then the printer_device and printer_copies parameters will also be respected.
xs_required	Split Report File Required	0	Determines if split files that have been successfully emailed or entered into the database should also be kept on disk (default = NOT xw_required)
xs_inrep	Input report name	CS04xx	Specifies the name of the input file (xls or rpt) to be used for this particular breakpoint. If undefined uses the value of xs_definrep or if that is not defined then the normal input file will be used for all breakpoints.
xs_report	Input report file name	Invoice_<format>.repx	Specifies the name of the report file within the current package (.rerx file) to be used for each split. Note that parameter substitution occurs so that

			different report formats can be used for different iterations.
xs_definrep	Default input report name	CS04def	See above.
xs_language	Language for split report texts	<:column_name>	Defined the language to be used for the multilingual texts in the split reports. Note this can evaluate to a different language for each split report.
xs_logall	Logging flag	1	Set to 1 to include log messages from all iterations of split reports. If 0 or not present only the first, last and every 1000th iteration is logged.
xs_distrep	Distribution input report name		If defined this report will be run against the distribution log dataset.
xs_password	Password applied to split report output	<id>	Causes the value of the “id” column to be applied as a password to pdf / xlsx output files (sets parameter xf_password to this value)
xw_required	Whole Report Required	0	Determines if whole/unsplit report is also to be produced (default=1)
xw_filetype	File Type of main output file	pdf	Any of the supported output file types for the given report type.
xw_language	Language for whole report texts	FR	Defined the language to be used for the multilingual texts in the whole report.
xw_report	Report file name	WholeReport.xlsx	Name of report file to be used to generate the whole/unsplit report. If not specified then the report file specified within the report package (.rerx) is used.

xw_load	Load sheet	1	Excelerator only. Determines if the whole report is to be loaded. The default value is 1 if xw_post is 0 and 0 otherwise.
xw_post	Post sheet	0	Excelerator only. Determines if the whole report is to be posted back. The default value is 0. Set it to 1 if the whole report is to be validated or to 2 if it is to be posted.
xw_password	Password applied to output of whole report	My Password	Causes the value of the “id” column to be applied as a password to pdf / xlsx output files (sets parameter xf_password to this value)

10.2. DOCUMENT ARCHIVING (XD_)

ParamID	Param name	Example value	Comments
xd_doctype	Document Type	IMAGE	The Document Type to be used in the Document Archive
xd_indexvals	Index Values	<client>,<apar_id>	A comma delimited list of index values that must correspond to the required index as defined on the Document Type.
xd_title	Document Title	Statement <period>	A title for the Document
xd_desc	Document Description	Statement for <period>	A description for the Document
xd_comments	Document Comments	As at <report_date>	A comment for the Document
xd_archive	Document Archive Name	XX	Archive name. This defaults to the current client which is the correct value in almost all installations. So this parameter is only needed if the document is to be placed in some other archive, perhaps a different client.

10.3. E-MAILING (XE_, XA_)

ParamID	Param name	Example value	Comments
xe_to	Email To	<email_address>	Defines the email address to which the output is to be sent, parameters will be substituted at each breakpoint. NB requires that the email address exist in the. xe_attid/value is probably more useful.
xe_cc	Email Cc	<cc_name>	Defines the email address to which the output is to be carbon copied. Parameters will be substituted at each breakpoint. NB requires that the email address exist in the data
xe_bcc	Email Bcc	<bcc_name>	Defines the email address to which the output is to be blind carbon copied. Parameters will be substituted at each breakpoint. NB requires that the email address exist in the data
xe_from	Email From	DoNotReply@xxx.com	Defines the email from address for sent mail (the user usually sees this email address)
xe_replyto	Email Reply To	ReplyHere@xxx.com	Defines the "Reply To" email address for sent mail (most systems will use this address if the user hits "reply"). Requires ABW 5.6.x or higher.
xe_subject	Email Subject	Statement <apar_id>	Defines the email subject
xe_text	Email Text	Text for customer <apar_id>	Defines the email text
xe_addrtype	Address Type	1	Defines the address type to be used (defaults to 1=general)

xe_attid	Address att id	A4	Attribute ID to which the address is attached
xe_attvalue	Address att value	<apar_id>	Attribute value for this break point. NB any string can be entered but should contain a parameter in Excelerator format that will evaluate to the appropriate value for each split. defaults to <xs_breakcol>
xe_addrid	Address ID		Explicit address id / link into agladdress (some standard ARWs already provide this information)
xe_except	Exception report required	0	Specifies if exception report containing items for which no email address was found is required. (default=1)
xe_reserve	Reserve email address	admin@company.com	Specifies an alternative e-mail address to use if the initial e-mail address causes an immediate error.
xe_error	Treat missing email address as error	1	Tells Report Engine to treat a missing email address as an error (default=0)
xa_file1 xa_file2…	File(s) to be attached	C:\temp\myfile.pdf C:\temp\anotherfile.pdf	Specifies the fully qualified name of a file to be attached to the emailed report output. xa_file2 will only be recognized if xa_file1 is defined etc. Report Engine does not impose a limit on the number of attachments although the underlying email system might.
xa_doctype1 xa_doctype2..	Document type(s) to be attached	ORDER INVOICE	Specifies the document type of the document(s) to be attached. xa_doctype2 will only be recognized if xa_doctype1 is defined etc. Report Engine does not impose a limit on the number of attachments although the underlying email system might.

xa_docindex1	Document index(s) to be attached	<client>,<apar_id>	Document indexes corresponding to the document types defined with xa_doctype.
xa_docindex2		<client>,<apar_id>,<seq>	

10.4. FILING (XF_)

ParamID	Param name	Example value	Comments
xf_file	File name for split report output	Statement<apar_id>	Defines where the split report output will be put. xf_file may specify a relative but not an absolute path. (see also xf_dir).
xf_password	Password to be applied to report output file	My_password	Only applicable to pdf / xlsx output files. This will affect split and whole report. Better to specify xw_password or xs_password and allow the system to set xf_password accordingly. Note: this must be “declared” in resql in order to take effect.
xf_pfda	Set this flag to create pdf file in pdfa (archive format	1	Only applicable to pdf files. If this value is non-zero then the pdf output file will be created in pdfa / archive format. Default = 0.
xf_title	Title to be applied to pdf file	Invoice	
xf_author	Author to be applied to pdf file	Company Name	
xf_subject	Subject to be applied to pdf file	Project Invoice <period>	
xf_jpgsize	Image quality in exported pdf files	10	Only applicable to pdf output files Set a value from 1 (minimum) to 100 (maximum) to control the size/quality of images in

			exported pdf files. (default is maximum=100)
xf_dir	Directory name for split report output	C:\Report Output	Specifies the root directory for split report output. If both xf_dir and xf_file are specified then the file will be written to a file named xf_file in directory xf_dir. Note that parameter substitution does NOT occur in xf_dir and the directory must already exist. (defaults to the current TEMP folder)
xf_exists	Split report file exists action	0	Defines the action to take when a split report file already exists. Values are: 0 – create a new file; 1 – stop; 2 – overwrite the file; 3 – skip to next file. The default value is 0.
xf_missing	Split report file is missing action	0	Defines the action to take when a split report file does not exist. Values are: 0 – create file; 1 – stop; 2 – skip to next file. The default value is 0.

10.5. FILING (XTRA REPORTS ONLY)

ParamID	Purpose	Example value	Comments
xf_open_password	Equivalent to xf_password.		If both are present, xf_password will prevail in order to keep backwards compatibilty. Controls opening of doc
xf_editing_fields	Export editable fields to pdf	1	If set then any fields with "Edit Options>Enabled = Yes" in the report will be editable in the output pdf report.

xf_never_embed	Prevent embedding of named fonts	“Arial;Times New Roman”	If present the named fonts will not be embedded in the output pdf file. By default, all used fonts will be embedded. Setting this parameter may reduce the file size considerably.
xf_permissions_password	Password applied to changing document properties, copying, printing…		if this password is not set, the following options won't be available. Default values is they're not explicitly set, are marked per option.
xf_printing_permissions	Sets printing permissions	2	0 = Not allowed (DEFAULT) 1 = Low resolution printing only 2 = High resolution printing enabled
xf_changing_permissions	Sets the level of changing permissions	3	0 = No change is allowed (DEFAULT) 1 = inserting, deleting and rotating pages 2 =filling in form fields and signing existing signature fields 3 = commenting, filling in form fields and signing existing signature fields 4 =anything except extracting pages
xf_enable_copying	Set to allow copy text of document	1	0 = not allowed to copy text (default) 1 = allowed to copy text
xf_enable_screenreaders	Allows text access for screenreaders	0	0 = allow access 1 = disallow access (default)

10.6. MISCELLANEOUS

ParamID	Param name	Example value	Comments
locale	Sets the locale / culture for the report	en-GB	Values may be either an integer representing the LCID of the desired culture or a string containing it's name
xr_required	Indicates whether or not to keep a copy of the output .rerx file in "Report Results"	0	Values are: 0 – do not keep the rerx file; 1 keep the rerx file. Default = 0

11. REPORT ENGINE SQL (RESQL)

11.1. CONCEPTS

Report Engine SQL (.resql) files are always embedded within Report Package Files (.rerx)

ReSql governs the process of extracting data from the database and/or an input .NET DataSet to create an output .NET DataSet. This output DataSet will later be rendered to create the actual report by one of the Reporter plug-ins (Excelerator, ARC, Worderator) which, itself, has no access to the database. Thus all the data for the report (but none of the formatting information) must be included in the DataSet.

11.2. GRAMMAR

ReSql files are plain text files. Lines in the resql file beginning with a full stop "." are command lines that give instructions directly to the ReSql processor. All the valid commands are listed below. All other lines contain SQL commands. These commands may be

- Regular SQL commands that will be executed against the database
- Unit4 Business World specific extensions beginning with names like agr_ also executed against the database
- Report Engine SQL commands prefixed with RE that will be parsed by Report Engine. Use this.tablename to select data from the dataset that is in the process of being created or data0.tablename to select from the initial/input dataset.

All types of SQL as well as the contents of the command lines obey the normal SQL syntax for quoting literals.

String literals are written with single quotes: 'my string'

Date literals are written 'yyyy-mm-dd' or 'yyyy-mm-dd hh:mm:ss'

Numeric literals always require a "." decimal separator irrespective of the current regional settings (as per SQL) e.g. 1 or 4.5 or -1.648e+5

Identifiers need not be quoted unless they contain spaces or other characters or begin with a numeric in which case they are quoted with double quotes or square brackets:

```
my_table_name
[my column name]
"my column alias"
[1a]
```

Whitespace (including new lines) is ignored in all SQL commands.

Tables with names beginning with ‘#’ are temporary and removed from the dataset when processing is complete.

11.3. PARAMETER SUBSTITUTION

Parameters and data fields may be referenced within .if statements and within SQL commands as follows:

`$?parameter1` will be replaced by the current value of parameter1.

`$:field1` will be replaced by the value of field1 from the current row of the table being looped over.

`$:table1.field1` will be replaced by the value of field1 from the last row that was added to table1.

`$$rowcount` will be replaced by the number of rows affected by the most recently executed SQL command.

`$$dbtype` will be replaced by “1” when connected to MS SQL Server, “2” when connected to Oracle or “0” otherwise

11.4. REFERENCE

11.4.1. .name

```
.name dataset_name
```

Sets the name that will be given to the DataSet that is created (DataSet.DataSetName). Normally, this is the first statement in the .resql file.

11.4.2. .parserLevel

```
.parserLevel {asql|database|native}
```

Sets the default parser level to be applied to SQL SELECT statements (this can be overridden by prefixing the individual statements if required). Note that "database" and "native" are synonymous. If not specified the parser level defaults to “native”.

11.4.3. .dbtimeout

```
.dbtimeout n
```

Sets the database timeout period to *n* seconds.

11.4.4. .declare

```
.declare parameter_name data_type default_value [?]
```

Declares / Creates a parameter of the given name type and default value. .declare statements must appear before any other statement (except .name, .*, .parsertype or .log) in the .resql file.

Parameter names may not contain spaces.

Valid data types are

string
int / int32 / int64
datetime
float / real / double

Examples

```
.declare my_string string 'hello world'

.declare my_int int 42

.declare my_date datetime '2013-06-12'

.declare current_time datetime 'now'
```

('now' is a special case which causes Report Engine to set the parameter value to the current time)

The optional “?” at the end of the statement causes the user to be prompted for the value of this parameter when the resql is executed in an interactive environment (Excel/Word Add-in or Report Studio). This can be very useful if the report is to be deployed in this way but if such a “prompting” parameter is encountered when running as part of a non-interactive process it will cause an the process to fail and stop immediately.

The default value may now also be an expression as per .set.

11.4.5. .set

```
.set parameter_name = value
```

Sets the value of the named parameter. The parameter must have been previously declared, otherwise an exception will be raised.   The supplied value must be able to be converted to the correct type for the parameter.

Parameters can be referenced in other statements as follows

```
.set my_number = 1
.set my_string = 'Hello World'

.set my_datetime = '2012-08-27 14:49:00'
.log Number = $?my_number, string = $?my_string, date = $?my_datetime
```

This causes the following to be written into the log file

```
Number = 1, string = Hello World, 2012-08-27 14:49:00
```

You may also use simple arithmetic expressions like this

```
.set i = 1

.set j = $?i + 1
```

The following operators are supported "+", "-", "*", "/" and "%" (modulo)

For string parameters only "+" (concatenate) is valid.

"%" (modulo) is only valid of integers.

The expression may also be a bracketed SELECT statement in which case the value of the first column in the first row will be used. E.g.

```
.set j = (SELECT COUNT(*) FROM acrclient)

Or

.set j = (RE SELECT x FROM this.my_table WHERE …)
```

11.4.6. .log

```
.log "text to appear in log file"
```

Causes the specified text to be written into the log file.

11.4.7. .* comment

```
.* this is a comment line
```

Any line beginning with .* is a comment and will be ignored

11.4.8. .#region / .#endregion

```
.#region region_name

.* some part of the resql file that will be shown or hidden in the editor

.exec CREATE TABLE …

.#endregion
```

These statements have no effect on the process of the file, they are ignored in the same way that comment lines are ignored. Their sole purpose is to allow the enclosed region of the resql to be shown or hidden in the ReSQL editor.

11.4.9. .if

```
.if condition
   block1
.elseif condition2
   block2
.else
   block3
.endif
```

If *condition* is true then all lines in block1 are executed and lines in block2 and block 3 are ignored. If *condition2* is true then block2 is executed otherwise block3 is executed.

Conditions take the form of

```
[!] value1 (=|!=|>|>=|<|<=) value2
```

normally at least one of the values is parameterized

e.g.

```
.if '$?client' = 'EN'

    .log "client is EN"
.else
    .log "client is not EN"

.endif
```

11.4.10. .query

```
.query table_name
    select_command

.endquery

Or

.query table_name select_command
```

Runs the sql command and creates a table named table_name in the DataSet containing the results.

`.query main` indicates that this is the main table that will be used to drive the report splitting if the report is to be split.

11.4.11. .splitQuery

```
     .splitQuery table_name
         select_command

     .endquery

     Or

     .splitQuery table_name select_command
```

This is exactly equivalent to “.query main” except that it gives you the possibility of splitting by a table named something other than “main”

11.4.12. .singleRowQuery

```
     .singleRowQuery table_name
         select_command

     .endSingleRowQuery

     Or

     .singleRowQuery table_name select_command

Or

     .singleRowQuery table_name
      (select_command1) [prefix1], (select_command2) [prefix2] [, ...]
     .endSingleRowQuery
```

Runs the sql command and creates a table named table_name in the DataSet containing the results.

Differs from .query in that only the first row will be included in the results and that the single row join syntax available.

11.4.13. .addParentRow / .addRow / .addFirstRow

```
     .addParentRow table_name
         select_command
     .endaddParentrow

Or

     .addParentRow table_name select_command
```

```
Or
```

```
     .addParentRow table_name
      (select_command1) [prefix1], (select_command2) [prefix2] [, ...]
     .endAddParentRow
```

.addFirstRow and .addRow are simply synonyms for .addParentRow

Runs the sql command and adds the resulting row to the specified table. If the table does not exist it is created. If it does exist the row is appended to the table. If the table exists but has a different schema then an exception is thrown. If more than one row is returned an exception is thrown.

If used within a .loop a relation will be created from the .loop table (child) to the added table (parent).

Note that in the case of .addRow you may join several select_commands using the third syntax. This is very useful to create a well-structured DataSet.

11.4.14. .addRows / .addChildRows

```
     .addRows table_name
        select_command
     .endAddRows
```

```
Or
```

```
     .addRows table_name select_command
```

.addChildRows is simply a synonym for .addRows

Runs the sql command and adds the results to the specified table. If the table does not exist it is created. If it does exist the rows are appended to the table. If the table exists but has a different schema then an exception is thrown.

If used within a .loop a relation will be created from the .added table (child) to the loop table (parent).

11.4.15. .loop / .group

```
.loop table_name
    .group column1
        [group block1]
        .group column2 [,column3 [, ...]]
            [group block2]
        .endgroup [column2 [,column3 [, ...]]]
    .endgroup [column1]
.endloop
```

Iterates through rows in specified table.

Statements in *group block1* are executed once for each time the value of column1 changes

Statements in *group block2* are executed once for each time the value of column1, column2 and column3 changes

Groups may be nested as deeply as desired and the enclosed blocks of statement may contain .if / .endif as well as .addRow and .addRows commands.

E.g.

```
.query main
    SELECT client, account FROM ...
.endquery

.loop main
    .group client
        .addRow clients
            SELECT client_name FROM acrclient WHERE client='$:client'
        .endaddrow
        .group account
            .addRows details
                SELECT description, amount FROM some transaction table...
                WHERE client='$:client' AND account='$:account'
            .endaddrow

        .endgroup

.endgroup
.endloop
```

In this example there the column 'client' will be added to the 'clients' table as well as the 'client_name' and a relation from clients.client (parent) to main.client (child) will be created.

Also the columns 'client' and 'account' will be added to the 'details' table and a relation from main.client,main.account (parent) to details.client,details.account (child) will be created.

NB. Using relations is a much more efficient way of saving the data as the header information is not repeated for every row. You can make use of these relations using the 'EXPAND' keyword when selecting data from the resulting dataset within an ARC or Excelerator report.

11.4.16. .relate

```
.relate table1.column1[,column2,…] table2.column1[,column2,…]
```

Creates a relation from table1 (the parent table) to table2 on the specified columns. This can be a more efficient way of achieving the same result as using a loop structure when it is possible to select all rows for the related table in one SELECT statement rather than using a .loop structure.

11.4.17. .exec

```
.exec command
```

Or

```
.exec

 command

.endexec
```

Executes any of the following dataset sql commands that update the current dataset but do not return a result set:

11.4.17.1. CREATE TABLE

```
CREATE TABLE name (column type [UNIQUE] [, column type [UNIQUE] [, ...]] )

CREATE TABLE name select_command
```

Where *select_command* may select from the database, the input dataset or the current dataset.

Valid types are

String

int / int32 / int64

datetime

float / real / double

Creates a table in the current dataset. N.B. this does not update the database in any way.

11.4.17.2. DROP TABLE

```
DROP TABLE name
```

Drops a table from the current dataset. N.B. this does not update the database in any way.

11.4.17.3. INSERT

```
INSERT INTO table [(column1, column2, column3)] VALUES (value1, value2, value3)

INSERT INTO table [(column1, column2, column3)] select_command
```

Where select_command may select from the database, the input dataset or the current dataset.

Inserts row(s) into the named table in the current dataset. N.B. this does not update the database in any way.

11.4.17.4. UPDATE

```
UPDATE table SET column1 = value1|(select_command1)

[, column2 = value2|(select_command2) [, ... ] ]

WHERE ...
```

Where select_command may select from the database, the input dataset or the current dataset.

Updates row(s) into the named table in the current dataset. N.B. this does not update the database in any way.

11.4.17.5. DELETE

```
DELETE FROM table WHERE ...
```

Deletes row(s) from the named table in the current dataset. N.B. this does not update the database in any way.

ReSql files only. Not applicable to reports.

11.4.18. .attachFile

```
    .attachFile file_name
```

Causes the specified file to be attached when the report output is distributed by email. This can be used to attach a different or multiple files to each email

11.4.19. .attachDocument

```
    .attachDocument client, doc_type, doc_index1, doc_index2 …
```

Causes the specified document from the Unit4 Business World document archive to be attached when the report output is distributed by email. This can be used to attach a different or multiple files to each email

11.5. BUSINESS WORLD SPECIFIC SELECT COMMANDS

In an ReSql file SQL command may either be regular SQL SELECT statements or one of the 'agr_' macros listed below:

11.5.1. agr_getaddress

```
agr_getaddress client, attribute_id, dimValue, addressType [, [sequenceNo]
[, language]]

agr_getaddress client, address_id
```

This will return a single row containing all available columns from table agladdress plus 3 additional columns:

- "formatted_address" containing the address formatted according to the Unit4 Business World defined rules.
- "formatted_address2" containing the same thing but with blank lines removed.
- “formatted_address3” same as “formatted_address2” but with commas instead of line breaks

If sequenceNo is -1 or missing the first matching row (lowest sequenceNo) will be returned.

If sequenceNo is -2, the last (highest sequenceNo) row will be returned.

11.5.2. agr_getdescription

```
agr_getdescription client, attribute_id, dimValue [, language ]
```

Returns a single row containing the description of the given attribute value.

11.5.3. agr_getDocument

```
agr_getdocument client, docType [, index1 [, index2 [, index3 ... ]]]
```

Returns a single row containing the specified document.

11.5.4. agr_getDocuments

```
agr_getdocuments client, docType [, index1 [, index2 [, index3 ... ]]]
```

Exactly like agr_getDocument except that it will return all matching documents if more than one exists.

11.5.5. agr_getText

```
    agr_gettext textType, variant [, language]
```

Returns a single row table containing the specified text.

e.g. to fetch the statement text for report CS04:

```
    agr_gettext '4', 1, '$?language'
```

11.5.6. agr_getSysConf

```
    agr_getsysconf [client], name1 [, name2 [, name3 ...]]
```

Returns a table with columns named 'name1', 'name2' etc and a single row containing the values of the corresponding System Configuration parameters.

11.5.7. agr_getTitles

```
    agr_gettitles reportName, language, *|[id1 [, id2 [, id3 ... ]]]
```

Returns a single row with one column for each title. E.g.

```
    .query titles
        agr_gettitles 'CS04', 'EN', '*'
    .endquery
```

will cause all the named titles for report CS04 plus the report header titles to be returned in English.

11.5.8. agr_getbrowser

```
    agr_getbrowser template_name [, param1=value1 [, param2=value2 [, param3=value3
    … ]]]
```

Runs the specified browser template with the given parameters and returns the results.

Parameters given must match those defined in the browser template. The internal names of columns followed by one of the suffixes below must be given like...

dim1_eq	for dim1 equal (or like)
dim1_ne	not equal (or not like)
dim1_ge	greater than or equal
dim1_gt	greater than
dim1_le	less than or equal
dim1_lt	less than
dim1_in	in list (format is dim1_in='value1,value2,value3')
dim1_ni	not in list

Parameters defined in the Browser “SearchC” screen marked as prompting must all be provided, others may be provided if desired.

Note that if the parameter is not given then the correct names of the required parameters are given as part of the error message.

11.5.9. agr_getbalance

```
agr_getbalance

SELECT “attname1” [, ”attname1_text” [, ”attname2” [, … ]]],

“value1” [, ”value2” [, …]]

FROM balanceTableName(period, client, language) [REVERSE] [NOZEROS]

RELATION “related_att1” “att1”, “related_att2” “att2”

TREE “tree_name1” [, ”tree_name2”]

WHERE “attname1” “criteria1” [AND “attname2” “criteria2” [AND … ]]

ORDER BY “attname1” [, ”attname2” [, … ]]
```

This statement has equivalent functionality to the Excelerator “query balance” construct. Indeed, “query balance” now generates an agr_getbalance statement that is then processed by the server. It is possible to see these generated statements in the log file which can be very helpful when migrating reports.

Note that the criteria appearing in the agr_getbalance statement follow the format of criteria strings entered in Excelerator reports rather than an SQL like format.

e.g. This excelerator query will generate the following agr_getbalance statement which may then be included directly in an resql file with parameter substitution etc.

query balance aggadp EN						
relation division, costc						
relation section, costc						
where section 10-20						
sort costc desc, account						
	division	section	costc	account	period	amount
columns	text division	text section	text costc	text account	text period	amount
detail						0

```
agr_getbalance SELECT "division","section","costc","account","period","amount"

FROM "aggadp"(0,'EN','EN')

RELATION "division" "costc","section" "costc"

WHERE "section" "10-20"

ORDER BY "costc desc","account"
```

12. DATASET SQL AVAILABLE TO REPORTS

Within an ARC or Excelerator report running against a DataSet the only way of retrieving data is with the Select command. This is a limited version of the full SQL SELECT syntax and is described below.

12.1. SELECT

The DataSet SELECT command is a limited dialect of the SQL SELECT command that operates on tables contained within a dataset. This is the only statement available within ARC or Excelerator reports when agr.UseDataSetOnly or SQL parser = DataSet is selected.

The syntax is as follows:

```
SELECT {*|column1 [[as] alias1]|literal_value [as] alias1 [,column2 [[as]
alias2] [,column3 ...]]}

FROM [EXPAND] table

WHERE condition1 [AND|OR condition2 [AND|OR condition3 [...]]]

{GROUP BY column1 [, column2 [, column3 ...]] |

ORDER BY column1 [asc|desc], column2 [asc|desc] [, column3 ...] }
```

Note that the FROM clause may contain only one table. The EXPAND keyword will cause the table to be de-normalized such at all columns from parent tables will be included as if they were located in the table itself.

Note also that there is no HAVING clause. The same result can be achieved with a WHERE clause applied to the result table

```
SELECT * FROM (SELECT ... FROM ... WHERE ... GROUP BY ...) WHERE ...
```

Conditions may be prefixed with NOT to negate their meaning and grouped together with round brackets to force the order of calculation. (Otherwise AND takes precedence over OR).

Valid conditions are:

```
[NOT] column =|<>|<|<=|>|>= literal_value|column

[NOT] column IN(literal1, literal2 [, literal3 [...]])

[NOT] column LIKE('pattern')
```

pattern may contain % or * as wildcard characters either at the beginning or end (or both) of the string. E.g. '*x' will match anything ending in an 'x', 'x*' will match anything beginning with an 'x', '*x*' will match anything containing an 'x'. 'x*x' is not permitted.

12.1.1. Identifiers

Identifiers (column names, aliases and table names) are normally not quoted or can be quoted with square brackets or double quotes to allow names starting with a numeric character or containing spaces. E.g.

```
column1
[1a]
[my column name]
"another column"
```

12.1.2. String Literals

String literals are quoted with single quotes. If a single quote character is required within the literal value then this must be escaped with a second single quote. E.g.

```
'my string literal value'
'O''Brian'
```

12.1.3. Date Literals

Date literals are written in yyyy-mm-dd [hh:mm:ss] format and quoted with single quotes. E.g.

```
'2013-06-11'
'2013-06-11 15:33'
'2013-06-11 15:33:55'
```

12.1.4. Integer Literals

Integer Literals are written simply as [+|-]n and are not quoted. E.g.

```
1
-32
+47
```

12.1.5. Real Number (amount) Literals

Real number (amount literals) are written [+|-]nnnn.nn and are not quoted. Note that as always in SQL the decimal point is always written with a full stop "." and without thousands separators irrespective of the current localization settings of the machine (i.e. always 1234.5 never 1,234.5 never 1.234,5). E.g.

```
1.3
-3.14159
+47.0
```

12.1.6. Boolean Literals

Boolean literals are written 1 or 0 and are not quoted. E.g.

```
0      (= false)
1      (= true)
```

12.1.7. UNION

```
 (RE SELECT ... FROM ...)

UNION

(agr_getaddress ...)
```

Brackets are mandatory in order to allow this operation to be performed.

This operation reflects actual UNION sql syntax, and therefore works well under following conditions:

- The two branches must have the same schema (same data types) and the same number of columns.
- The resultant column name should have the name of column from first branch.
- You can have as many UNION as you like, respecting above conditions.

12.1.8. Functions

12.1.8.1. FORMAT

```
SELECT FORMAT('{0:yyyy-mm-dd},{1},{2:0.00}', my_date, my_string, my_amount) x FROM this.#table
```

This function provides access to the full power of the C# string.Format() function to allow you to format strings as required. This is especially useful in conjunction with the csv reporter as it allows creation of fixed width files etc.

12.1.8.2. GET_TITLES_ROW

```
SELECT GET_TITLES_ROW(language [, defaultLanguage [, name_column, title_column, language_column]])
FROM this.#titles
```

This function pivots the "*title_column*" in the given table such that one column is returned for each existing value of "*name_column*" for the given "*language*". Note that one column is returned for every distinct value of "*name_column*" in the table whatever the language. This is so that the same columns will always be returned for all languages even if rows are "missing" from the table. The intended use is to implement the "Benelux" or other similar multilingual titles scheme as in this example…

```
.query #titles SELECT label_name name, label_text title, language FROM ultbnclabeldetail WHERE code =
'$?label_name'


.loop purchaseOrder

    .group language

        .addrow t

            RE SELECT GET_TITLES_ROW('$:language') FROM this.#titles

        .endaddrow

    .endgroup

.endloop
```

Brackets are mandatory in order to allow this operation to be performed.

This operation reflects actual UNION sql syntax, and therefore works well under following conditions:

- The two branches must have the same schema (same data types) and the same number of columns.
- The resultant column name should have the name of column from first branch.
- You can have as many UNION as you like, respecting above conditions.

13. SECURITY MODEL

This section discusses the security issues involved in running Report Engine reports and how these should be dealt with.

End users should be given access to relevant data that they are entitled to see and not to any other data. It is, of course, particularly important that end users are not given the opportunity to make arbitrary changes to SQL statements contained in a report and then re-execute the report. On the other hand, a report a report designer user will sometimes need full read access to large parts of the database in order to develop the report in the first place. You have to trust some of the people some of the time but you definitely should not trust all the people all the time nor even some of the people all of the time.

Report Engine applies security to the Data Extraction process not to the report rendering. I.e. once the data is extracted from the database and delivered to the user it is theirs to render/display/analyse as they see fit absolutely no security rules are applied at the Report Rendering stage. Note that this implies that the report designer should not create an ReSQL file that extracts any data that the user is not permitted to see even if this data is not displayed in the report.

It is important to understand the different categories of report that exist as different privileges are required in order to run them.

In Report package-based reports Data Extraction is driven by the contents of the ReSql file that they contain. In “traditional” Excelerator and ARC reports the Data Extraction is driven by the SQL SELECT statements or “query balance” or “query agresso …” constructs that they contain. From the security perspective there are 3 classes of Data Extraction statements/constructs.

Raw SQL SELECT statements: These could return any data from anywhere in the database and so must be the most tightly controlled.

“query balance …” (Excelerator) / agr_getbalance (ReSQL): These could in principle return any data stored in an Unit4 Business World Balance Table and any data related to such data via “relations” or “trees”. It is important to note that Unit4 Business World User based security is not applied at this point. So these are not as open/dangerous as raw SQL but it is still important for developers of reports to control the parameters that can be applied to such queries to avoid giving access to data that the user should not see.

“query agresso” (Excelerator) / agr_getbrowser (ReSQL): These constructs extract data via the Unit4 Business World Query Engine which enforces the Unit4 Business World security and so such reports can be run by anyone who has a valid Unit4 Business World login.

13.1. REPORTS FROM SECURE LOCATIONS

By “secure location” we mean that the report has been read from the “Cusomised Reports” folder on the Unit4 Business World report server machine or from the Unit4 Business World database itself where it was previously imported.

All types of report can be run by any user if the report is from a secure location and the user has access to the launching point for the report (i.e. the report ordering screen or browser template against which the report is imported). This is the same rule as is applied to all types of report and customisation software. It is therefore the responsibility of the report developer who places the report in a secure location to make sure that the report will not give the user access to information that they are not entitled to see. In practise, this is usually rather easy for the report designer to implement.

Access to the “Customised Reports” folder is controlled at the operating system level. No doubt it is well understood that only fully trusted users should be given this access for specific purposes. We must stress that a user having read ‘R’ access to the “Report Architect” menu point in Unit4 Business World has the right to import reports into the database which is exactly equivalent to allowing them to put report files into “Customised Reports” and so this too, should be controlled.

13.2. REPORTS FROM UN-SECURED LOCATIONS:

All other locations are un-secured. In particular if a user has an Excelerator report / xlsx file open on their desktop this is an un-secured location. The user could, if they so wished, make any change to the report and re-execute it. Therefore, access to data via the Report Engine web service is controlled. This is driven by the access that the user has to

Privilege	Powers
Read access to "Report Architect"	Can execute any report from an un-secured location giving read access to the entire database. Can import any report against screens and browser templates and make them available to other less privileged users.
Create access to "Report Architect"	Can execute Excelerator "query balance" reports from un-secured locations giving read access to all data contained in all Unit4 Business World balance tables. But cannot run reports containing raw SQL statements from un-secured locations.
All users	Can run but not change any report to which they have access from secure location. Can get data only via "query agresso"/agr_getbrowser statements only from an un-secured location. I.e. they will not be able to run any report containing "query balance" or raw SQL statements.
Write access to "Customised Reports" folder	Can cause any report to be made available to less privileged users on the report server.

Report Engine does also support making an ODBC connection directly to any database. If this feature is used it is entirely the responsibility of the database administrator to control the access that is given to the database at the ODBC / Database level.