The SeETL

Meta Data Dictionary

User Guide

Version 3.2.00
1st January 2019
Instant Business Intelligence
info@instantBI.com

Table of Contents

1. CHANGE CONTROL LOG 5

2. AUDIENCE 6

3. INTRODUCTION 7

4. TABLES IN THE SEETL METADATA DICTIONARY 8
4.1. Summary of Tables 8
4.2. Details of Tables 14
4.2.1. ctl_audit_table 14
4.2.2. ctl_message_table 15
4.2.3. ctl_last_key_used 16
4.2.4. ctl_aggregation_control 17
4.2.5. ctl_dim_table_key_definitions 19
4.2.6. ctl_dim_table_type2_col_defs 22
4.2.7. ctl_dim_table_load_control 23
4.2.7.1. Guidelines on In Memory Array Allocations 25
4.2.8. ctl_batch_control 28
4.2.9. ctl_month_control 30
4.2.10. ctl_codes_lookup 31
4.2.11. ctl_mapping_ss 32
4.2.12. ctl_file_cycle_control 36
4.2.13. ctl_sequence_nums 37
4.2.14. ctl_batch_pre_req 39
4.2.15. ctl_batch_run_log 42
4.2.16. ctl_column_defs 43
4.2.17. ctl_commands 44
4.2.18. ctl_datastage_parms 46
4.2.19. SeETL DDL Generator 47
4.2.19.1. ctl_ddl_gen_table 47
4.2.19.2. ctl_ddl_gen_columns 49
4.2.20. ctl_proc_grp_pre_req 51
4.2.21. ctl_proc_grp_run_log 53
4.2.22. ctl_proc_run_log 54
4.2.23. ctl_process_commands 55
4.2.24. ctl_data_types 57
4.2.25. ctl_src_file_defs 58
4.2.26. ctl_mapping_doc_infa 61
4.2.26.1. ctl_mapping_doc_infa_01 61
4.2.26.2. ctl_mapping_doc_infa_02 64
4.2.27. ctl_segmentation_control 65
4.2.28. ctl_dw_views_doc 66
4.2.28.1. ctl_dw_views_doc_01 66
4.2.28.2. ctl_dw_views_doc_02 66
4.2.28.3. ctl_dw_views_doc_03 67
4.2.28.4. ctl_dw_views_doc_04 68
4.2.28.5. ctl_dw_views_doc_05 69
4.2.29. ctl_data_model_joins 71
4.2.30. ctl_dw_sps_doc 72
4.2.30.1. ctl_dw_sps_doc_01 72
4.2.30.2. ctl_dw_sps_doc_02 73
4.2.30.3. ctl_dw_sps_doc_03 73
4.2.30.4. ctl_dw_sps_doc_04 74

4.2.30.5. ctl_dw_sps_doc_05 75
4.2.30.6. ctl_dw_sps_doc_06 77
4.2.30.7. ctl_dw_sps_doc_07 78
4.2.30.8. ctl_dw_sps_doc_08 79
4.2.30.9. ctl_dw_sps_doc_09 80
4.2.30.10. ctl_dw_sps_doc_10 81
4.2.31. ctl_dw_other_views_doc 82
4.2.32. ctl_dw_compound_views_doc 83
4.2.33. ctl_dw_sql_stmts 84
4.2.34. ctl_dw_sql_stmt_log 86
4.2.35. Data Profiling Worksheet 87
4.2.35.1. ctl_dw_dp_row_counts 87
4.2.35.2. ctl_dw_dp_col_counts_01 88
4.2.35.3. ctl_dw_dp_col_counts_02 89
4.2.35.4. ctl_dw_dp_col_counts_03 90
4.2.35.5. ctl_dw_dp_col_counts_04 91
4.2.36. ctl_dw_tables_doc 93
4.2.36.1. ctl_dw_tables_doc_01 93
4.2.36.2. ctl_dw_tables_doc_02 94
4.2.37. ctl_dw_count_sql_doc 96
4.2.37.1. ctl_dw_count_sql_doc_01 97
4.2.37.2. ctl_dw_count_sql_doc_02 97
4.2.38. ctl_dw_runstats_doc 98
4.2.38.1. ctl_dw_runstats_doc_01 98
4.2.38.2. ctl_dw_runstats_doc_02 99
4.2.39. ctl_dw_ctas_doc 100
4.2.39.1. ctl_dw_ctas_doc_01 100
4.2.39.2. ctl_dw_ctas_doc_02 101
4.2.40. ctl_dw_create_index_doc 102
4.2.40.1. ctl_dw_create_index_doc_01 102
4.2.40.2. ctl_dw_create_index_doc_02 103
4.2.41. ctl_dw_create_storage_doc 104
4.2.41.1. ctl_dw_create_storage_doc_01 104
4.2.41.2. ctl_dw_create_storage_doc_02 105
4.2.42. ctl_dw_permissions_doc 106
4.2.42.1. ctl_dw_permissions_doc_01 106
4.2.42.2. ctl_dw_permissions_doc_02 107
4.2.43. ctl_dw_generic_object_doc 108
4.2.43.1. ctl_dw_generic_object_doc_01 108
4.2.43.2. ctl_dw_generic_object_doc_02 109
4.2.44. ctl_dw_report_doc 110
4.2.44.1. ctl_dw_report_doc_01 110
4.2.44.2. ctl_dw_report_doc_02 111
4.2.44.3. ctl_dw_report_doc_03 112
4.2.44.4. ctl_dw_report_doc_04 113
4.2.44.5. ctl_dw_report_doc_05 114
4.2.45. ctl_dw_project_plan_doc 116
4.2.45.1. ctl_dw_project_plan_doc_01 116
4.2.45.2. ctl_dw_project_plan_doc_02 117
4.2.45.3. ctl_dw_project_plan_doc_03 118
4.2.45.4. ctl_dw_project_plan_doc_04 119
4.2.45.5. ctl_dw_project_plan_doc_05 121
4.2.45.6. ctl_dw_project_plan_doc_06 123
4.2.45.7. ctl_dw_project_plan_doc_07 124
4.2.45.8. ctl_dw_project_plan_doc_08 125
4.2.46. ctl_dw_biz_reqts_doc 126
4.2.46.1. ctl_dw_biz_reqts_doc_01 126
4.2.46.2. ctl_dw_biz_reqts_doc_02 129
4.2.46.3. ctl_dw_biz_reqts_doc_03 132
4.2.46.4. ctl_dw_biz_reqts_doc_04 135
4.2.46.5. ctl_dw_biz_reqts_doc_05 138
4.2.46.6. ctl_dw_biz_reqts_doc_06 141
4.2.46.7. ctl_dw_biz_reqts_doc_07 144

4.2.46.8. ctl_dw_biz_reqts_doc_51 147
4.2.46.9. ctl_dw_biz_reqts_doc_52 150
4.2.46.10. ctl_dw_biz_reqts_doc_53 153
4.2.46.11. ctl_dw_biz_reqts_doc_54 156
4.2.46.12. ctl_dw_biz_reqts_doc_55 159
4.2.46.13. ctl_proc_grp_start_run_log 162
4.2.46.14. ctl_dw_delta_definitions 163
4.2.46.15. ctl_dw_nav_max_entry_nos 163
4.2.46.16. ctl_dw_nav_max_timestamps 163
4.2.46.17. ctl_dw_nav_min_entry_nos 163
4.2.46.18. ctl_dw_nav_min_timestamps 163
4.2.46.19. ctl_dw_pre_mappings_01 163
4.2.46.20. ctl_src_table_defs 163
4.2.46.21. ctl_src_column_defs 163
4.2.46.22. ctl_src_index_defs 164
4.2.46.23. ctl_src_to_stg_defs 164

1. CHANGE CONTROL LOG

#	Date	Name	Description
3.1.00	1/01/2012	IBI Dev	The MetaData Dictionary Documentation was first released on 1st January 2012. This release is because it is seen as beneficial to place a single description of all tables in the dictionary into one document. The tables in the dictionary have been documented next to the functions they supported. This made sense when the dictionary was very simple. Now the dictionary is much more comprehensive and contains many more fields it has been decided to continue to maintain the documentation in the existing places as well as to break it out and create one large document that documents all tables in one place. This will save the person new to SeETL from searching multiple places to find documentation about a table. In many cases the documentation will be redundantly cut and paste into this document.
3.1.01	1/01/2013	IBI Dev	Extensions for 3.1.01.
3.1.02	1/01/2014	IBI Dev	Extensions for 3.1.02.
3.2.00	1/01/2019	IBI Dev	Extensions for 3.2.00.

2. AUDIENCE

The intended audiences for the SeETL Meta Data Dictionary are:

- Technical developers working on a Star Schema Data Warehouse project using SeETL.
- Technical developers implementing BI4ALL Data Models.
- IT managers considering the purchase of BI4ALL Data Models or SeETL.

3. INTRODUCTION

In the early days of SeETL development there was very little information stored in the dictionary. Just some rudimentary tables to assist building tables and views. The idea at the time was to NOT have any dictionary because these became more complex and more bloated and in many cases 'got in the way' of productivity.

It was not until the invention of reading the workbook with vb.net code and placing the contents of the workbook into the dictionary that it made sense to start building out the dictionary.

In the early days of the dictionary build out we tried very hard to keep it 'lean and mean'. We did not want to clutter up the dictionary with all manner of very useful things that would then need maintenance and support.

All this changed in 2009 when working at client. We invented a new way to generate SQL as the finished ETL subsystem. This went on to be used by another client as the sole ETL tool in 2010. As a result of these projects and others it became clear that if there was something of a 'build out' of the dictionary massively greater function could be delivered.

So in the first half of 2011 we determined to build out the dictionary with MANY more tables and offer this extended dictionary to our clients.

Should you, as a client or prospective client, have some more ideas about what can be put into this dictionary?

Feel free to contact us by our web site www.instantbi.com.

Please note:

As of SeETL 3.2.00 we are no longer documenting the contents of the dictionary in complete detail for a number of reasons.

1. ***SeETL is now free and we all know documentation is not read very much any more. So "complaints" about "the documentation could be more complete" do not really carry any weight any more. It's 2019.***

   The new documentation is video and we will be releasing most of our comments on videos on our video channel.

2. ***We feel it is pretty obvious to any competent DW Architect what the contents of the tables are and what they do.***

   We intend the software to be used for free by competent DW Architects.

3. ***We intend to document all tables in SeETL itself and to provide documentation of the columns in that workbook. This is a low priority task that will be done "one day". In the meantime there are more pressing features to develop and we will work more on creating video demonstrations which are the new way to document and promote a free product like SeETL.***

4. TABLES IN THE SEETL METADATA DICTIONARY

The SeETL Metadata Dictionary can be delivered onto any database platform that is considered viable for building data warehouses.

The SQL that will be included in this documentation is purely the SQL Sever SQL. We have versions for other databases. However, everyone who reads this will have access to an SQL Server database and will be familiar with the SQL Server create table syntax.

Like all dictionaries this Meta Data Dictionary is merely a set of tables designed to support the SeETL and to make the development of Data Warehouses faster and easier. The naming standards are very simply to understand. Fields that are prefixed with pk_ are part of the primary key.

Whether the DBA at the client decides to add constraints or indexes or anything else to tables is up to the DBA. We deliver the tables as is without indexes (etc) and there is little problem with this because the tables are so small in general.

Now we have data profiling tables we might need to build some indexes on them. But this is a trivial activity.

4.1. Summary of Tables

The current list of tables in the Meta Data Dictionary and their meaning is as follows.

Table Name	Table Description
ctl_audit_table	This table contains detailed audit record for all programs that execute in the C++ version of the SeETL ETL Engine.
ctl_message_table	This table contains that messages that are written by the C++ version of the SeETL ETL Engine.
ctl_last_key_used	This table contains the last key used for dimension tables that are processed by SeETL. The keys can be used in both the C++ and SQL versions of the ETL Engine. Some clients use 'max' of the keys in the dimension table. However, if this is done then the client rules out the use of multi-level data.
ctl_aggregation_control	SeETL has a unique aggregation control facility that is better than anything else that has ever been done. This table controls the level of aggregation in fact tables merely be storing entries in a table.
ctl_dim_table_key_definitions	This table stores the definitions of what fields will be used to construct the string key of a dimension table.
ctl_dim_table_type2_col_defs	This table stores the definitions of what fields will be used to determine the type 2 changes to dimension tables.
ctl_dim_table_load_control	SeETL has very fine control of the loading of dimension tables into memory. This was required prior to the memory mapped IO function being introduced into SeETL. Once the memory mapped IO function was introduced virtually all clients turn this on all the time. This has become more so the case as memory prices have dropped and 64 bit memory became pervasively available.
ctl_batch_control	This table stores the batch numbers and batch dates of the batch processing of the scheduler. It also notes which batch is 'in progress' should there be one in progress.
ctl_month_control	This table stores useful variables such as the first day of last month, first day if this month, first day of next month as well as the high and low dates that will be used by a client.
ctl_codes_lookup	All clients have sets of 'codes' that translate into descriptions. This table is simply a place to store such codes and descriptions. It is independent of the BI4ALL Data Models.
ctl_mapping_ss	The heart of SeETL DesignTime is the mapping worksheet of the workbook. In this worksheet the client can define source to target mappings for virtually an entire data warehouse. Usually there are two or three very small workbooks as well. However, the mapping worksheet is the 'source code' for the ETL subsystem. It evolved from prior versions of mapping spreadsheets that were

	merely documentation that then needed to be converted to 'source code'. This table stores all fields from the mapping worksheet in a versioned table. The versioning is manual in that the version must be changed inside the spreadsheet itself. This versioned copy of the workbook inside a table can then be published via the reports that are supplied with the product to provide accurate documentation of the mappings to the business users.
ctl_file_cycle_control	SeETL contains very reasonable audit capabilities that are 'designed into the software' and data models. One of these is the 'file cycle number'. This table maintains the number of rows to be processed from staging into target for all mappings along with the batch number and the file cycle number of the file. These two numbers are then tagged onto all rows that flow through so as to give the Tech Support a very good chance of recovery should any really severe problem arise. The philosophy of SeETL recovery for databases is to fix the problem rather than to simply restore the database. The databases are becoming too large to sensibly 'restore'. The 'restore' can now take days instead of hours due to volumes of data being placed into data warehouses. Hence we provide a number of features to help Tech Support perform fixes in preference to 'restore'.
ctl_sequence_nums	This table retains sequence numbers that are placed onto fact tables when those tables are being processed by the C++ version of the SeETL ETL Engine. A key feature of this table is that the C++ Engine requests as range of keys and these keys are then granted to the calling process in a locked form. This allows for large files such as call detail records to be split and processed simultaneously and to guarantee there are no overlapping keys.
ctl_column_defs	This table is a documentation table that documents all the table/view names, the columns and the data types. It is very useful as a documentation tool to be able to query the data types of columns in views very easily and quickly. It is also used in the reporting of data models.
ctl_datastage_parms	SeETL has a utility that can start DataStage jobs from inside the scheduler. Therefore, if a client has DataStage and also runs other processes the scheduler can be used as the 'master scheduler' for all processes. We actually found on one project that because of the greater control over load balancing we were ale to cut the batch run time of DataStage jobs by 30% merely by using our scheduler. When using the utility to start DataStage jobs the parameters for a job can be stored in this table and therefore the same job can be run at different 'levels' of development merely by pointing to different tables for different parameters for the different levels. All very useful if a client wants to integrate DataStage into a broader environment.
ctl_ddl_gen_columns	Early versions of SeETL had the ability to generate DLL for tables that was fairly basic. This has been superseded by new features in the SeETL Workbook. However the old utilities remain available. This table stores column definitions for tables to be generated.
ctl_ddl_gen_table	Early versions of SeETL had the ability to generate DLL for tables that was fairly basic. This has been superseded by new features in the SeETL Workbook. However the old utilities remain available. This table stores table definitions for tables to be generated.
ctl_batch_pre_req	This table contains the pre-requisites to be met prior to a batch being able to be started. This is usually a 'signal file' that is to arrive to tell the batch that the files to be processed have arrived, or are starting to arrive, as the case may be.
ctl_batch_run_log	This table contains the log records at the batch level for the scheduler. All batches are logged.
ctl_commands	This table contains the commands that will be processed by the scheduler.
ctl_proc_grp_pre_req	This table stores the pre-requisites for process groups to run in the scheduler.
ctl_proc_grp_run_log	This table stores the run log for the process groups.
ctl_proc_run_log	This table stores the run log for processes.
ctl_process_commands	This table stores what are called 'process commands'. These commands have substitution parameters which is an enhancement on prior versions of the commands table for the scheduler.

ctl_data_types	This table is a documentation table to translate the ODBC number that is returned for a data type which is then interpreted as the ODBC data type to the real data type of the underlying database. It is simply a translation documentation table to convert numbers to the data types that the Tech Support or business users will understand.
ctl_src_file_defs	This table contains the documentation of all the source files that may be considered for being placed into the data warehouse.
ctl_mapping_doc_infa_01	SeETL is used to develop prototypes for all our clients. However, many of our clients have Informatica as a 'standard' and the ETL must eventually be deployed into Informatica. This is usually done by sending the specifications for the ETL subsystem off to an 'Indian Programming House'. To do this documents must be exchanged to instruct the programmers as to what they must write. This table (and the next one) document the Informatica specific portions of the mappings and link across to the mapping spreadsheet table so as to be the source for PDF reports that can be sent to the offshore programming company to produce the informatica jobs. This has proven very successful on a number of projects.
ctl_mapping_doc_infa_02	This table contains the second portion of data for the specification of Informatica mappings.
ctl_segmentation_control	This table contains control information for the Segmentation Engine.
ctl_dw_views_doc_01	The SeETL workbook has been upgraded to allow the documentation of the ***BI4ALL*** Data Models in the workbook. These data models are then loaded into the dictionary to be made available for query and reporting purposes. This table is the first table in the series of tables used to document ***BI4ALL Data Models***.
ctl_dw_views_doc_02	This table is the second table in the series of tables used to document ***BI4ALL Data Models***.
ctl_dw_views_doc_03	This table is the third table in the series of tables used to document ***BI4ALL Data Models***.
ctl_dw_views_doc_04	This table is the fourth table in the series of tables used to document ***BI4ALL Data Models***.
ctl_dw_views_doc_05	This table is the fifth table in the series of tables used to document ***BI4ALL Data Models***.
ctl_data_model_joins	This table documents the joins between tables in the ***BI4ALL Data Models***. These joins can be written to such tools as Business Object Designer. These joins are also used to create Referential Integrity Constraints in the BIDS Documentation Database.
ctl_dw_sps_doc_01	The SeETL Workbook has been upgraded to support the translation and generation of Stored Procedures as well as to store documentation for Stored Procedures. We see Stored Procedures are going to take on more and more importance into the future. The reason being that the queries and analysis that is being performed on large data warehouse is becoming more and more complex and this is defeating the ability of people to express via GUI tools to present in reports. Tools such as Business Objects, Cognos, Microstrategy use GUI interfaces to express the SQL that is to be generated. And these interfaces are now 'getting in the way' the same way that ETL tool GUI interfaces have always gotten in the way of building complex ETL. This set of 10 tables documents the various aspects of Stored Procedure translation and generation and the stored procedures themselves. Refer to the detailed section of this document for more details on each table.
ctl_dw_sps_doc_02	This is the second table in the series for documenting the various aspects of Stored Procedure translation and generation and the stored procedures themselves.
ctl_dw_sps_doc_03	This is the third table in the series for documenting the various aspects of Stored Procedure translation and generation and the stored procedures themselves.

ctl_dw_sps_doc_04	This is the fourth table in the series for documenting the various aspects of Stored Procedure translation and generation and the stored procedures themselves.
ctl_dw_sps_doc_05	This is the fifth table in the series for documenting the various aspects of Stored Procedure translation and generation and the stored procedures themselves.
ctl_dw_sps_doc_06	This is the sixth table in the series for documenting the various aspects of Stored Procedure translation and generation and the stored procedures themselves.
ctl_dw_sps_doc_07	This is the seventh table in the series for documenting the various aspects of Stored Procedure translation and generation and the stored procedures themselves.
ctl_dw_sps_doc_08	This is the eighth table in the series for documenting the various aspects of Stored Procedure translation and generation and the stored procedures themselves.
ctl_dw_sps_doc_09	This is the ninth table in the series for documenting the various aspects of Stored Procedure translation and generation and the stored procedures themselves.
ctl_dw_sps_doc_10	This is the tenth table in the series for documenting the various aspects of Stored Procedure translation and generation and the stored procedures themselves.
ctl_dw_other_views_doc	This table stores the source data for the generation of what are known as 'other views' in an IBI Implementation. In every implementation that are some views that are not part of the data model or part of the views automatically generated by the workbook. They are placed into a text document called 'Other Views'. However, should the client wish to document these views into the workbook and load the workbook into the dictionary so as to ensure that all source for views is within the workbook this option has been provided.
ctl_dw_compound_views_doc	Very occasionally views are needed that are compound views. This is especially so for 'union views'. Since Union Views have a difference structure to all other 'create view' statements and they are so rare in a data warehouse ETL subsystem they were placed into a separate workbook. This table stores the source data for the input to the generation of Union Views. It can be used to link back to the underlying tables to assist establish data linearage.
ctl_dw_sql_stmts	The SeETL Runtime has been enhanced to allow the execution of SQL Statements that are actually stored in a database rather than in a file system. This improves the stability, auditability, security and readability of the ETL subsystem. This table contains the SQL statements loaded into the database including full history of all such statements.
ctl_dw_sql_stmt_log	This table contains the log of all the statements executed by the ETL subsystem.
ctl_dw_dp_row_counts	SeETL has been upgraded to perform Data Profiling functions. These are very useful functions to calculated information about data both at the exploration stage of developing the data warehouse as well as runtime stage. This table stores row counts of tables.
ctl_dw_dp_col_counts_01	This table stores the second set of data with respect to Data Profiling functions.
ctl_dw_dp_col_counts_02	This table stores the third set of data with respect to Data Profiling functions.
ctl_dw_dp_col_counts_03	This table stores the fourth set of data with respect to Data Profiling functions.
ctl_dw_dp_col_counts_04	This table stores the fifth set of data with respect to Data Profiling functions.
ctl_dw_tables_doc_01	SeETL has been upgraded to allow the definition and generation of 'create table' DDL to be stored in the SeETL Workbook. The definitions are relatively simple and allow for the use of different databases for the deployment and documentation databases. Free format text allows for the unstructured portions of the create table statements as well as the wide differences in the create table statements from the various database vendors. This table contains the first set of data with respect to the create table functions.
ctl_dw_tables_doc_02	This table contains the second set of data with respect to the create table functions.
ctl_dw_count_sql_doc_01	This table contains the directory that generated SQL to count input rows will be

	placed in.
ctl_dw_count_sql_doc_02	This table contains the source data that will be used to generate SQL Count statements as part of the audit process for SeETL ETL subsystems.
ctl_dw_runstats_doc_01	This table contains the directory that runstats SQL will be placed in.
ctl_dw_runstats_doc_02	This table contains the runstats SQL along with documentation columns so that the runstats SQL can be easily found with queries.
ctl_dw_ctas_doc_01	This table contains the directory that create table as SQL will be placed in.
ctl_dw_ctas_doc_02	This table contains the create table as SQL along with documentation columns so that the create table as SQL can be easily found with queries.
ctl_dw_create_index_doc_01	This table contains the directory that create index SQL will be placed in.
ctl_dw_create_index_doc_02	This table contains the create index SQL along with documentation columns so that the create index SQL can be easily found with queries.
ctl_dw_create_storage_doc_01	This table contains the directory that create storage object SQL will be placed in.
ctl_dw_create_storage_doc_02	This table contains the create table as SQL along with documentation columns so that the create storage object SQL can be easily found with queries.
ctl_dw_permissions_doc_01	This table contains the directory that create permissions SQL will be placed in.
ctl_dw_permissions_doc_02	This table contains the create table as SQL along with documentation columns so that the create permissions SQL can be easily found with queries.
ctl_dw_generic_object_doc_01	This table contains the directory that create generic object SQL will be placed in.
ctl_dw_generic_object_doc_02	This table contains the create table as SQL along with documentation columns so that the create generic object SQL can be easily found with queries.
ctl_dw_report_doc_01	The SeETL Workbook has been upgraded to provide the ability to record a wide array of information about reports that are being developed or deployed onto the Data Warehouse. The creation of good data linearage tools has been something that has really been missing from the BI vendors toolkit. The ability to document reports across multiple query tools has also been missing. This set of tables is our first effort to document reports for clients in such a way that they are useful and provide the ability to link the field on a report all the way back to the data model and the source fields from which the field on the report was derived. This is the first table in the series of tables to document reports.
ctl_dw_report_doc_02	This is the second table in the series of tables to document reports.
ctl_dw_report_doc_03	This is the third table in the series of tables to document reports.
ctl_dw_report_doc_04	This is the fourth table in the series of tables to document reports.
ctl_dw_report_doc_05	This is the fifth table in the series of tables to document reports.
ctl_dw_project_plan_doc_01	All IBI Projects require a Project Plan to be created so that the client can properly monitor project progress. Over the years we have used MS Project on most of our projects. MS Project has proven to be singularly difficult in managing projects that are highly iterative in nature. As a result we are releasing a workbook that allows us to manage our projects using this as a tool for the project. It is expected that only a small percentage of our clients will use this tool. However, we feel it will be more useful than MS Project and other project management tools. This is the first table in the series of tables to document project plans.
ctl_dw_project_plan_doc_02	This is the second table in the series of tables to document project plans.
ctl_dw_project_plan_doc_03	This is the third table in the series of tables to document project plans.
ctl_dw_project_plan_doc_04	This is the fourth table in the series of tables to document project plans.
ctl_dw_project_plan_doc_05	This is the fifth table in the series of tables to document project plans.
ctl_dw_project_plan_doc_06	This is the sixth table in the series of tables to document project plans.
ctl_dw_project_plan_doc_07	This is the seventh table in the series of tables to document project plans.
ctl_dw_project_plan_doc_08	This is the eighth table in the series of tables to document project plans.
ctl_dw_biz_reqts_doc_01	SeETL can now load Business Requirements into the metadata dictionary.

	This is the first table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_02	This is the second table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_03	This is the third table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_04	This is the fourth table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_05	This is the fifth table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_06	This is the sixth table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_07	This is the seventh table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_51	This is the eighth table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_52	This is the ninth table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_53	This is the tenth table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_54	This is the eleventh table in the series of tables to document Business Requirements.
ctl_dw_biz_reqts_doc_55	This is the twelfth table in the series of tables to document Business Requirements.
ctl_proc_grp_start_run_log	This is a control table that logs the start of each process group.

4.2. Details of Tables

In the SeETL Runtime C++ Engine all operations are audited as closely as possible.

As can be seen it logs the program that ran, the start and stop timestamps and the number of rows processed.

The Field Name, Integer Total and Dollar Total hash totals were never actually implemented. We included these fields and offered to clients that we would hash total fields on the rows processed. All clients we talked to took the position that this was not necessary and so the idea was dropped. In the 8 years we have been using this audit facility at clients we have not had one insist on putting this function into the C++ version of the ETL engine.

4.2.1. ctl_audit_table

The ctl_audit_table contains the following fields.

Field Name	Description
program_name	The program name that issued the write of the audit/control record to the database.
field_name	The name of the field that was audited or a short message to tell you what file record count was reported on this record.
total_rows	The total number of rows processed via this file for the run.
integer_total	The hash total for an integer field that was audited as a 'critical field'
dollar_total	The hash total for a dollar field that was audited as a 'critical field'.
started_tstamp	The date time that the program was started using a timestamp.
ended_tstamp	The date time that the audit record was written to the database using a timestamp.

Example DDL for the table is as follows:

```
create table dbo.ctl_audit_table
    (  program_name      varchar     (0256)       not null   default 'unknown'
    ,  field_name        varchar     (0256)       not null   default 'unknown'
    ,  total_rows        integer                  not null   default 0
    ,  integer_total     integer                  not null   default 0
    ,  dollar_total      money                    not null   default 0
    ,  started_tstamp    datetime      not null   default current_timestamp
    ,  ended_tstamp      datetime      not null   default current_timestamp
    )
;
```

4.2.2. ctl_message_table

The SeETL Runtime C++ Engine issues a large number of error messages. These Error Messages are at three levels. Information, Warning and Severe. Obviously Information Messages are copious and meant to inform the IT Support merely that something happened, not necessarily something bad.

Therefore there are Information Messages for such things as Program Starting and Program Ending.

The Warning Messages are there to warn the Technical Support of a situation that has been observed that looks like a problem but processing can still continue.

The Severe Messages generally mean that the program is about to halt because it has found a situation that is totally unacceptable and processing can not safely continue.

The ctl_message_table contains the following fields.

Field Name	Description
program_name	The program name that issued the write of the error message to the database.
function_name	The name of the function inside the program that issued the write of the error message to the database.
message_number	The error message number. This can be used to look up error message documentation and to report to the technical support for the product.
message_severity	The severity of the message. There are three levels: • I – Information. The message is provided for information only and can generally be ignored. Examples of Information Messages are 'Program Starting.' 'Program Ending.' • W – Warning. A condition has been found that is unusual and unexpected but processing is not stopped. All warning messages should be investigated. • S – Severe. A condition has been found that is unexpected and processing will be stopped. Examples are where the ODBC connection parameters do not result in a connection being established, input or output tables not being found etc.
message_text	The text of the message. Messages are issued to be as descriptive as possible.
date_time	The system date time that the error message was issued.

Example DDL for the table is as follows:

```
create table dbo.ctl_message_table
    (  program_name      varchar     (0256)    not null default 'unknown'
    ,  function_name     varchar     (0256)    not null default 'unknown'
    ,  message_number    varchar     (0010)    not null default 'unknown'
    ,  message_severity  varchar     (0001)    not null default 'unknown'
    ,  message_text      varchar     (4000)    not null default 'unknown'
    ,  date_time         datetime               not null default current_timestamp,
    )
;
```

4.2.3. ctl_last_key_used

The ctl_last_key_used table is used to store the last key used for each dimension in the same database as the dimension table updates occur.

The ctl_last_key_used table contains the following fields.

Field Name	Description
pk_int_key	This is the primary key of the table. It is an integer that you can enter via some tool such as MS Access. You only ever need to enter it once per dimension table so no front end has been developed.
dim_table	The name of the 'dimension table' for which this record stores the last key used. This can be any valid database table or view name. It is recommended that it is the name of a view. Examples are 'time_dim', 'customer_dim' etc.
last_key_used	This will store the last key used for a particular dimension table. You only ever enter the key at start-up time for the data warehouse and it is not recommended that it is ever changed. Integer keys will be allocated sequentially for each dimension table. We prefer not to use database specific sequence keys as they can often impede unload/load activities on tables by insisting on providing new keys for records being inserted.

Example DDL for the table is as follows:

```
create table dbo.ctl_last_key_used
    (  pk_int_key       integer                 not null primary key
    ,  dim_table        varchar     (255)       not null default 'unknown'
    ,  last_key_used    integer                 not null default 0
    )
;
```

4.2.4. ctl_aggregation_control

One of the major features of SeETL [RT] is the ability to generate new aggregates, without any program changes, merely a parameter change in this table.

The ctl_aggregation_control table contains the following fields.

Field Name	Description
pk_aggregate_number	This is an integer primary key to this table. It is actually passed through to the summary fact table to provide the ability to partition the summary fact table by the individual summary. It can also be specified in a view to define the fact table as a separate table to end user tools such as business objects and microstrategy.
run_type	The type of run for this aggregation. Summaries which you want to occur for every aggregation run of the fact table should be marked as 'always'. Aggregation levels that you want to generate just once, perhaps for special analysis, you should mark as 'oneoff'.
fact_table_name	The name of the detailed level fact table that you want the summary to be generated from. An example is f_order_fact. The summary table will be assumed to be 'f_order_fact_summary'. Thus you should choose naming standards for fact tables that allow the suffix '_summary' to be added to the table name.
number_of_dimensions	The number of dimensions in this fact table. This field is used to select the correct number of columns from the level_dimension_n fields. For example if there are 4 dimensions, only the first 4 level_dimension fields will be read as significant.
level_dimension_1	The level of the dimension to be used in this summary level. For example 0 means 'detail' level, 1 means 'level1', 2 means 'level2' etc. You have one level field for each dimension for each aggregation for the fact table. This means that the level key for that level will be used in the summary fact table. You can have many levels of aggregation and hence for the time dimension you can construct levels at daily, weekly, monthly, quarterly, annual and total levels if you choose to.
level_dimension_2	The same explanations as level_dimension_1 except this is for the second dimension on the fact table.
etc.	
level_dimension_50	The same explanations as level_dimension_1 except this is for the fiftieth dimension on the fact table.

Please note there are a maximum of 50 dimensions allowed for any single fact table. If anyone has a need to go over this please let us know. The code is easy to change.

Example DDL for the table is as follows:

```
create table dbo.ctl_aggregation_control
    (  pk_aggregate_number     integer          not null  primary key
    ,  run_type                varchar    (20)  not null  default 'always'
    ,  fact_table_name         varchar (256)   not null  default 'unknown'
    ,  number_of_dimensions    integer          not null  default 0
    ,  level_dimension_1       integer          not null  default 0
    ,  level_dimension_2       integer          not null  default 0
    ,  level_dimension_3       integer          not null  default 0
    ,  level_dimension_4       integer          not null  default 0
    ,  level_dimension_5       integer          not null  default 0
    ,  level_dimension_6       integer          not null  default 0
    ,  level_dimension_7       integer          not null  default 0
    ,  level_dimension_8       integer          not null  default 0
    ,  level_dimension_9       integer          not null  default 0
    ,  level_dimension_10      integer          not null  default 0
    .
    .
    .
    ,  level_dimension_45      integer          not null  default 0
    ,  level_dimension_46      integer          not null  default 0
    ,  level_dimension_47      integer          not null  default 0
    ,  level_dimension_48      integer          not null  default 0
    ,  level_dimension_49      integer          not null  default 0
    ,  level_dimension_50      integer          not null  default 0
    )
;
```

Example rows for this table are as follows:

pk aggregate number	run type	fact table name	number of dimensions	level dimension 1	level dimension 2
1	always	f_order_fact	2	2	1
2	always	f_order_fact	2	5	0
3	always	f_order_fact	2	1	5

These rows mean that the f_order_fact_summary table will contain 3 aggregate levels. There are only 2 dimensions in the fact table.

The first aggregate will summarise the first dimension table at the key level 2 and the second dimension at the key level 2.

The second aggregate will summarise the first dimension table at the key level 5 and the second dimension at the detail level key.

The third aggregate will summarise the first dimension table at the key level 1 and the second dimension at the key level 5.

How these keys get allocated will be explained below in ctl_dim_table_key_definitions.

4.2.5. ctl_dim_table_key_definitions

One of the major features of SeETL [RT] is the ability to generate multiple levels of aggregates depending on information fed into the dimension table. These levels in the dimension tables can then be used to generate levels in the summary fact tables. In all other solutions we have ever seen, this must be done in code. A new aggregate level being made available in the dimension table means someone has to make a change to some code somewhere. No more!!

The ctl_dim_table_key_definitions table contains the following fields.

Field Name	Description
pk_int_key	This is an integer key to the table that you must enter.
dim_table_name	This is the dimension table name. For example time_dim, customer_dim, product_dim etc.
dim_type	This is the 'type' of the dimension, type 1 or 2. Valid values are 1 or 2.
agg_level	This is the aggregate level that the set of concatenated columns will represent. Valid values are 0-9. 0 means 'detail' and 9 means 'total'. In between they are known as 'level1', 'level2', 'level3' etc. Note that there is no need for the levels to actually roll up into each other. For example level 0 could be day, level 2 could be week and level 3 could be month. And weeks do not roll up into months.
num_columns_joined	This is the number of columns that are to be joined together to form the string character key for this level. If you enter a lower number than fields you want to join SeETL[RT] will only join the first n columns. If you enter a higher number than the number of column names you provide it will only join on the number of column names available. However it is highly recommended that the number of columns joined accurately reflects the number of column names you actually want to join. The maximum is 50 joined columns.
column_names	This is the set of column names you want to join together to form the character key to be used at this level. Every unique combination of these characters will be allocated an integer key and placed into the dimension table marked as being at that level. If you join non character columns (such as integers) they will be converted to character strings and then concatenated. Note that trailing blanks in an input field will also be truncated. The concatenated string of characters is limited to 255 characters. This seems more than enough since we have never used more than 150 characters previously. The concatenation character is the '+' character. Note that a column that is concatenated as part of the key that is null is allocated the internal character string 'NULL' and this string is placed onto the dimension record. This means that there is a restriction. It is recommended that columns used for dimension table keys that can be null should not have the real value of 'NULL' as a possible value.

Example DDL for the table is as follows:

```
create table dbo.ctl_dim_table_key_definitions
    (  pk_int_key             integer                  not null primary key
    ,  dim_table_name         varchar        (256)     not null default 'unknown'
    ,  dim_type               integer                  not null default 0
    ,  agg_level              integer                  not null default 0
    ,  num_columns_joined     integer                  not null default 0
    ,  column_names           varchar        (4000)    not null default 'unknown'
    )
;
```

Example rows for this table are as follows:

pk int key	dim table name	dim type	agg level	num columns joined	column names
0	customer_dim	1	0	1	cust_code
1	customer_dim	1	1	1	cust_household
2	customer_dim	1	2	2	cust_sex+cust_geocode
3	customer_dim	1	3	1	directive_none
4	customer_dim	1	4	1	directive_none
5	customer_dim	1	5	1	directive_none
6	customer_dim	1	6	1	directive_none
7	customer_dim	1	7	1	directive_none
8	customer_dim	1	8	1	directive_none
9	customer_dim	1	9	1	Directive_total
10	time_dim	1	0	1	day_field
11	time_dim	1	1	2	week_number+year_field
12	time_dim	1	2	2	month_number+year_field
13	time_dim	1	3	1	directive_none
14	time_dim	1	4	1	directive_none
15	time_dim	1	5	1	directive_none
16	time_dim	1	6	1	directive_none
17	time_dim	1	7	1	directive_none
18	time_dim	1	8	1	directive_none
19	time_dim	1	9	1	directive_total

The data in these rows has the following meaning.

Both customer_dim and time_dim have 1 detail level and 9 aggregate levels. You do not need to have 9 aggregate levels. However, if you want to leave spaces for possibly having more levels in the future you must enter the rows for each level you might want to have at some point in the future. SeETL [RT] assumes that the first time it reads this table it is reading the maximum number of levels you will ever want for this dimension. Increasing the number of levels at a later date is not tested and not supported.

Aggregate level '0' means the 'detail' level.

The text 'directive_none' has a special meaning. It means that this level of aggregate is reserved for future use but is not currently implemented.

The text 'directive_total' has a special meaning. It means that this level of aggregate is the highest level of aggregate and means total for that dimension. Hence, when you want to ignore a dimension in a summary you can set it to the total level and all rows in the summary level will have the same key.

The concatenation character '+' has special meaning in that it breaks the column names. The code looks at the '+' sign in this table as the delimiter of the field names. Naturally '+' is not allowed as a character in the table column name.

The concatenation of week_number+year_field means that when the dimension table is loaded these two fields will be joined together and the dimension table will be searched to see if this combination of week and year has ever been loaded. If not, it will allocate a key and insert the record thus creating a unique character string for week level and an integer key that can be used in the fact table to join to the dimension table record.

This processing is for type 1 dimensions and for the keys for type 2 dimensions. However, type 2 dimension tables have a further requirement to close out records and insert new records when a field that is not necessarily part of the key changes. Naturally, any field that changes in the key will insert a new record if it does not exist. To handle type 2 dimension tables another control table is required. It is discussed next.

4.2.6. ctl_dim_table_type2_col_defs

To support type 2 dimension table changes we must record the combination of fields that must be checked for change to determine if the existing dimension table record should be closed out and a new type 2 record should be inserted. The ctl_dim_table_type2_col_defs table supports this requirement.

The ctl_dim_table_type2_col_defs table contains the following fields.

Field Name	Description
pk_int_key	This is an integer key to the table that you must enter.
dim_table_name	This is the dimension table name. For example time_dim, customer_dim, product_dim etc.
num_columns_to_test	This is the number of columns that are to be tested to see if they have changed when compared to the current dimension table record. There is a limit of 50 fields.
column_names	This is the set of column names you want to test for change. Each column's data value on the incoming record will be compared with the value on the current record for the dimension. The comparison is null sensitive so if the values are both equal or they are both null the record will be considered not to have changed. The delimiter is again the '+' character.

Example DDL for the table is as follows:

```
create table dbo.ctl_dim_table_type2_col_defs
    (  pk_int_key             integer                 not null primary key
    ,  dim_table_name         varchar        (256)    not null default 'unknown'
    ,  num_columns_to_test    integer                 not null default 0
    ,  column_names           varchar        (4000)   not null default 'unknown'
    )
;
```

An example row for this table is as follows:

pk int key	dim table name	num columns joined	column names
1	customer_dim2	3	cust_addr_1+cust_addr_2+cust_phone

The data in this rows has the following meaning.

The customer dimension 2 is a type 2 dimension table. The fields to check (at the detailed level only) for change are cust_addr_1, cust_addr_2 and cust_phone. If any of these fields have changed then close out the current dimension record and open a new dimension record. The source of the dates for record closure and opening is discussed below.

4.2.7. ctl_dim_table_load_control

Fact table attribution processes have a 'need for speed'. There are two Fact Table Attribution programs provided with SeETL [RT]. The attribution process can consume very large amounts of processing power and run for very long elapsed times.

One of the standard ways to speed it up is to trade memory for speed. That is load the dimension tables into memory and search them there rather than read them through the database which is at least 10 times slower per read. This loading of dimension tables into memory is facilitated by the ctl_dim_table_load_control table.

The ctl_dim_table_load_control table contains the following fields.

Field Name	Description
pk_int_key	This is the integer key to this record.
fact_table_name	This is the name of the fact table for which this record specifies the dimension table load option. The fact table name can also be set to '*ALL*' to specify that the dimension table named in dim_table_name should be loaded as specified in the load_control_option field for all fact tables.
dim_table_name	This is the name of the dimension table for which this record specifies the dimension table load option. The reason that both the fact table and the dimension table are specified is so that you have the option of loading dimension tables into memory for specific fact tables. The larger fact tables will benefit more from loading dimension tables into memory.
load_control_option	Determines whether the dimension table will be loaded into memory and searched using a binary search or read from the database. Valid values are: 1. Read the table from the database. 2. Read the table from a 100 element array. If the row is not found then read it and put it into the array. (Note. This option operates exactly as Option 1 at this time. With memory mapped IO implemented this option may not be implemented in the future.) 3. Load the table into memory and read the dimension table from memory using a binary search. 4. Load the table into a memory mapped file backed by the system paging file for windows 2000 and by /tmp files on unix. If a dimension table is loaded into a memory mapped for by one attribution process (AT01/2) it is available for access by all attribution processes. Memory mapped IO has been added to provide 'unlimited scalability'. That is, large volumes of memory can be used to load one single copy of as many dimension tables as the user would like and these images are shared across as many processors as the user would like to use.
dimension_table_group	This field can be defined to be any string, though we do recommend 'GROUP1', 'GROUP2', GROUPn' etc. With the introduction of memory mapped IO we also introduced the ability to run many instances of CTLU012 at the same time and to load many dimension tables at one time. This is made possible by specifying groups of dimension tables to load into memory mapped files. The dimension tables that are to be loaded in a single group are defined using this field. They are loaded by CTLU012 using the `LoadDimensionTableGroup` parameter. By using this field it is possible for the user to gain the maximum amount of control over the loading of dimension tables into memory mapped files enabling the most efficient use of the machine resources to achieve maximum throughput of the overall batch while using the minimum amount of real memory to maintain copies of the dimension tables in the memory of the machine.

Example DDL for the table is as follows:

```
create table dbo.ctl_dim_table_load_control
    (  pk_int_key            integer                  not null primary key
    ,  fact_table_name       varchar        (256)     not null default 'unknown'
    ,  dim_table_name        varchar        (256)     not null default 'unknown'
    ,  load_control_option   integer                  not null default 0
    ,  dimension_table_group varchar        (256)     not null default 'unknown'
    )
;
```

Example rows for this table are as follows:

pk int key	fact table name	dim table name	load control option
0	f_order_fact	customer_dim	1
1	f_order_fact	time_dim	2
2	f_order_fact	product_dim	3
3	f_order_fact	product_dim	4

The data in these rows has the following meaning.

For the f_order_fact attribution process CTLAT01/02 will:

1. Read customer_dim and time_dim from the database
2. Load product_dim into in an in memory array and binary search the array when looking up product keys.
3. Load vendor_dim into a memory mapped in memory array and binary search that array. The vendor_dim table will also be available for any other instance of CTLAT01/02 to share while product_dim is private to the CTLAT01/02 process that loaded it.
4. Note that no record in this table for a fact table/dimension table combination is equivalent to specifying a load control option of 1, read the table from the database.

4.2.7.1. Guidelines on In Memory Array Allocations

The use of the in memory arrays is important enough to provide some advice and guidance on their use.

The allocation of these arrays has been built to be as efficient as possible in the use of memory so that it is possible to load the greatest amount of data from dimension tables into memory.

The dimension character string key field for each record only has as much memory allocated as the length of the actual string on the database record, plus a (char *) pointer to point to the string.

This innovation allows:

1. The dimension tables to have dimension character string keys defined as 255 character strings rather than some fixed limit. Having longer character strings for some dimensions is particularly useful.

2. For character strings of significant variation in length to be used in the one dimension table. This is very useful for multiple levels being built into dimension tables.

3. For the minimum use of memory for any particular set of character keys stored in a dimension table.

The integer keys are allocated as one large array that contains (number of records in the dimension table x number of integer keys per record). This single array is then navigated to find the set of keys required when the character string key is found.

The search mechanism is a binary search. Though it would be faster to use a hash search we have not done so. Should customers require hash searching of the dimension tables it can be implemented in a later release.

Calculating Memory Consumption

When determining which dimension tables to load into memory you should consider the following information which gives an indication of memory consumption.

Character keys memory requirement = (Number of rows x 4bytes) + Actual volume of string data.

Integer keys memory requirement = (Number of rows x Number of integer keys/row x sizeof (integer) bytes).

All memory for all dimensions must be able to be allocated for the entire life of any instance of CTLAT01/02.

For example if you load a customer dimension which has 9 aggregate levels and 1 million rows with an average character string length of 50 bytes you will require memory as defined below.

Character keys memory requirement	=	(1000000x4)+(1000000x51)
	=	4MB + 51MB
	=	55MB
Integer keys memory requirement	=	1000000x9x4
	=	36MB
Total Memory that will be allocated	=	91MB

If four instances of CTLAT01 are run at the same time and they all load the customer dimension into their own address space then the memory requirement will soon become quite significant. In general, now that memory mapped IO is available, large dimension tables should be loaded into shared memory and small tables should be loaded into private memory. Now that memory mapped IO is available only dimension tables which present keys in excess of **approximately 1.5GB (windows)** should need to be read directly from the database.

For 'flat' data models with character keys and only one integer key associated with them the average kind of row size one would expect is 30 bytes. **And 1.5 GB gives about 50M key rows**. This is enough for any company planning on running their DW on windows. On unix systems this approximate limit of 1.5GB is removed.

We will be impressed if any customers of SeETL [RT] have dimension tables with 1.5GB of keys!!

Memory Mapped IO

Memory mapped IO was added to the windows 2000 version of SeETL [RT] in version 1.6.1 and the unix version of SeETL [RT] in version 2.1. The memory mapped IO processing is separately licensable functionality for SeETL [RT] in the windows executable license of the software. It is 'always available' in the source code license version. The main purpose of introducing memory mapped IO is to remove the need to allocate large amounts of memory for multiple instances of the Attribution Program (AT01/02).

As described, memory mapped IO will load **one** copy of the dimension table into a memory mapped file and all Attribution processes will access that one copy saving large amounts of memory.

However, the user will benefit from understanding the various ways in which memory mapped IO can be used when setting up the batch process for SeETL [RT] in order to set up the batch stream most effectively.

There are two ways to have a dimension table loaded into memory:

1. Using the Load Dimension Tables into Memory Maps Utility (U012)

   or

2. Allow the Attribution process to load the dimension table into memory mapped IO.

These two methods have their advantages and disadvantages. If it is at all possible, it is ***highly recommended*** that the user use the Load Dimension Tables into Memory Maps Utility is used in ***all*** cases. However, it is acknowledged that it is possible SeETL [RT] might one day be used by a customer who has too much data for even one copy in memory to be in memory at the same time on a windows machine. And there is now a solution to this problem as well.

Using the Load Dimension Tables into Memory Maps Utility

The Load Dimension Tables into Memory Maps Utility scans the ctl_dim_table_load_control table and loads all dimension tables marked as being required to be loaded into memory mapped IO when it starts up. It does this whether those tables will actually be used by any Attribution programs in the current batch or not. The Load Dimension Tables into Memory Maps Utility does not know which tables **will** be used in the current batch just that the dimension tables should be loaded into memory.

The utility was written this way to allow SeETL [RT] to run with the minimum number of processes remaining resident during the batch processing. It would have been easy to write the utility so that one program only loaded one dimension table. This would also run a little faster because the loading of the dimension tables into memory could be parallelised. However, it would leave one process per dimension table running on the operating system for the duration of the Attribution processing. That could be a lot of processes for a long time. So it was decided to write the utility as just one program that loaded **all** dimension tables.

To do this there were features added to allow the Load Dimension Tables into Memory Maps Utility to be started before any Attribution process can start and to be stopped at the end of the batch. See the documentation for the Load Dimension Tables into Memory Maps Utility to see how this is implemented.

It is expected that no SeETL [RT] customer will ever have so many dimension tables that are so large that they cannot all be loaded into memory at the same time. However, it is possible, so a solution is provided.

Allow the Attribution Process to Load the Dimension Table into Memory Maps.

If the customer does not have the memory required to load all dimension tables they want to load into memory for the duration of the Attribution processing the Attribution programs contain a feature which will allow them to load the dimension table into memory mapped IO as and when the table is required. This means that only those tables that are required are loaded into memory. This further reduces the memory requirement for Attribution processing.

However, there is a cost associated with this approach. The user should be aware that for a memory mapped file to remain resident in memory a process must be using it. If no process is using it then it is discarded and must be loaded from the database again before it can be used. So a badly designed batch stream may re-load dimension tables a number of times.

Further, there is a level of communication required between processes to be able to know when a dimension table has been loaded into memory mapped IO. Because this situation is considered to be unlikely and there is an easy solution to the problem there is currently only minimal support for these communication requirements.

Those customers who have bought the source code can refer to the source code and the extensive comments in the source code to see exactly how this works.

What happens is as follows:

1. The Attribution Process detects that the dimension table is going to be used and detects that it should be loaded into a memory mapped file.

2. It asks the operating system if any such table is already loaded into a memory mapped file.

3. If not it reads the table to determine the size of the memory mapped table it needs to allocate.

4. It then allocates the memory mapped file and then re-reads the dimension table and loads it into memory. [1]

5. It then makes the memory mapped file available to other processes.

Of course, if two Allocation Processes arrive at the processor for starting at around the same time it is possible that they will both ask the question 'Is this dimension table loaded into memory?' at almost the same time and both get the answer 'no'. They would then ***both*** go ahead and read the dimension table twice and then both try to allocate a memory mapped file. One will allocate the file and the other one will get the answer 'memory mapped file already exists'.

In this case there will be a waste of reading through the dimension table twice for the second process. No harm is done and the processing works.

However, if the user is really unlucky and the second process has very bad timing it might try to create the memory mapped file, get the answer that it is already there, and try to use it, while the first process is still loading it. There are currently no semaphores implemented to stop this unlikely but remotely possible situation.

To avoid this remote possibility the user should be aware of this situation and should make ***absolutely sure*** that there is enough time for the first attribution process to load the dimension table into memory before starting the second attribution process. If any number of users start to have this problem semaphores will be introduced into SeETL [RT].

If the first Attribution Process finishes before the second Attribution process the memory mapped files will remain in memory. In fact, the memory mapped files will remain in memory until the last Attribution Process has terminated.

Please note. This option is not supported on unix platforms as it is not expected that SeETL [RT] will ever be used on a large unix machine by a client so large that the dimension table keys cannot be loaded into the memory mapped io of the unix machine. In such cases we recommend 'buy some more memory'. We have only implemented this option to compensate for the limited memory supported by windows 2000+.

[1] To load a dimension table into a memory mapped file requires 2 reads of the complete dimension rather than just the one read of the dimension table required to load it into normal memory. This is because when loading the table into normal memory the memory for each string can be 'malloced' when the string is read. When reading into memory mapped files it is much faster to calculate the full size of the memory mapped file required before starting to load it.

4.2.8. ctl_batch_control

Those of us who have been around for a while remember that batches are processed on effective dates and that the system date is a poor guide as to the correct date to use on a record. So it is with type 2 dimension tables for the date_from and date_to columns. For example, if a data warehouse load fails on a Friday night and it will not be fixed until Monday the batches from Friday, Saturday and Sunday need to be processed. These batches actually contain changes from different days. Therefore it should be possible to place the correct date of change onto the type 2 dimension records. This is facilitated with the ctl_batch_control table.

The ctl_batch_control table contains the following fields.

Field Name	Description
pk_batch_number	This is the number of the batch. It is a primary key for this table.
batch_date	This is the batch processing date for this batch. If batch processing has been delayed as in the example above this date can be changed to tell SeETL [RT] the processing date. As far as SeETL [RT] is concerned this date is the real date that applies to records being passed into it. SeETL [RT] uses batch_date – 1 day as the closing date for changed type 2 dimension records and batch date as the opening date for changed type 2 dimension records.
batch_complete_flag	This is a flag 0 or 1 to indicate that the batch is complete. When it is set to 0 the batch is in progress. SeETL [RT] looks for the row with the batch complete flag set to 0 to find the batch date for the current batch. It is a severe error for more than 1 row to have a batch complete flag set to 0. When the batch is complete the flag is set to 1. The first utility job (CTLBM01) that starts up each cycle checks that there is no row on the table with the batch complete flag set to 0 before inserting the new row. The last utility job that runs prior to completion (CTLBM02) updates the flag to 1 to indicate successful processing.

Example DDL for the table is as follows:

```
create table dbo.ctl_batch_control
    (  pk_batch_number                          integer    not null
    ,  batch_date                               datetime    not null
    ,  batch_complete_flag                      integer    not null
    ,  constraint        pk_ctl_batch_control    primary key
       ( pk_batch_number
       )
    )
;
```

Example rows for this table are as follows:

pk batch number	batch date	batch complete flag
1	2011-01-02	1
2	2011-01-03	1
3	2011-01-04	0

The data in these rows has the following meaning.

Batch 1 complete successfully on 2011-01-02.
Batch 2 complete successfully on 2011-01-03.
Batch 3 has not completed successfully. It was started on 2011-01-04.

4.2.9. ctl_month_control

Some typical issues that often come up are:

- What dates are considered the first day of the month?
- What dates are used as default dates when you do not want to store nulls in date fields?

These issues are catered for with the ctl_month_control table.

The ctl_month_control table contains the following fields.

Field Name	Description
pk_int_key	This is an integer primary key. There is only one record in this table so it can be any number you like.
last_month	This is the date that is considered to be the first day of the last month.
this_month	This is the date that is considered to be the first day of this month.
next_month	This is the date that is considered to be the first day of next month.
low_value_date	This is the date that is considered to be the default low date.
high_value_date	This is the date that is considered to be the default high date.

Example DDL for the table is as follows:

```
create table dbo.ctl_month_control
    (  pk_int_key                          integer     not null
    ,  last_month                          datetime    not null
    ,  this_month                          datetime    not null
    ,  next_month                          datetime    not null
    ,  low_value_date                      datetime    not null
    ,  high_value_date                     datetime    not null
    ,  constraint     pk_ctl_month_control  primary key
       ( pk_int_key
       )
    )
;
```

An example row for this table is as follows:

pk int key	last month	this month	next month	low value date	high value date
1	2011-01-01	2011-02-01	2011-03-01	1900-01-01	9999-12-31

The data in this row has the following meaning.

The first day of last month is 2011-01-01, the first date of this month is 2011-02-01, the first day of next month is 2011-03-01. The default low value date is 1900-01-01. (Note that none of these fields are used in version 1.0. However they are there for future code already planned.)

The default high value date is 9999-12-31. This date is placed in the ‘date_to’ column of type 2 dimension records when they are inserted to say that the data is current and stretches out into the future.

4.2.10.ctl_codes_lookup

One of the things we come across again and again is the storage of lots of codes in a data warehouse. It has happened so often we have created a simple table to store them. This table is not used in SeETL [RT], however you might find this little piece of advice handy.

The ctl_codes_lookup table contains the following fields.

Field Name	Description
pk_code_name	This is a name given to the code itself. For example it might be 'PolicyStatus' for the code table that records policy statuses.
pk_date_from	The date from which the code is valid.
pk_date_to	The date to which the code is valid.
pk_current_flag	A flag to say this is the current record. This table can record changes to meanings of codes over time.
code_value	The code value itself. For example for policy Statuses this might be '-1' meaning closed.
code_sdesc	A short description of what the code means. This is the value of the code that is likely to be printed on reports or put onto screens.
code_sdesc	A long description of what the code means. This is the value of the code that is likely to be looked up by people for a detailed note on what the code means.

Example DDL for the table is as follows:

```
create table dbo.ctl_codes_lookup
    (  pk_code_name                          varchar (30)    not null
    ,  pk_date_from                          datetime        not null
    ,  pk_date_to                            datetime        not null
    ,  pk_current_flag                       integer         not null
    ,  code_value                            varchar (10)    not null
    ,  code_sdesc                            varchar (15)    not null
    ,  code_sdesc                            varchar (255)   not null
    ,  constraint     pk_ctl_codes_lookup    primary key
       ( pk_code_name
       , pk_date_from
       , pk_date_to
       , pk_current_flag
       )
    )
;
```

4.2.11.ctl_mapping_ss

The table ctl_mapping_ss is the table that the SeETL [DT] Spreadsheet is stored in to enable analysis and reports to be written on the SeETL [DT] Spreadsheet. Reports may be written on the SeETL [DT] Spreadsheet information in any tools that can read this table. The values of the columns are defined in detail inside the spreadsheet.

This ability to write reports directly against the mapping spreadsheet that is known to be 100% consistent with the implemented ETL code is a HUGE breakthrough for ETL Architects. This is a 'world first'.

Field Name	Description
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_mapping_row	A generated sequence number so that the mapping row primary is made to be unique.
Use_flag_1	This is set to tell the generator if the column is to be used in the mapping process. There are now two use flags to indicate different uses of the field being mapped.
Row_Type	The type of row is set as follows: • **M** - 'M'apping. The source column is moved to the target column with no change. • **D** - 'D'erived. The valid sql in the Source Column row will be placed into the view 'as is'. You may also place double quotes around the valid sql if the excel spreadsheet is interpreting the sql to mean something. • **F** - 'F'rom clause. The text in source column name will be passed to the database as a from clause. • **W** - 'W'here clause. The text in the source column name will be passed to the database as a where clause. • **G** - 'G'roup By - The text in the source column name will be passed to the database as a group by clause. • **O** – 'O'rder By – The text in the source column name will be passed to the database as an order by clause. Note that in from, where, group by and order by clauses the text for the sql statement goes into column name so that the new reports can use table name as a parameter for selecting data for reports.
Source_Name	Not currently used. However, we plan to use the staging name as the name of the ODBC connection defined in the processing worksheet.
Source_Owner	The owner of the source table. It is used in the qualification of sql generated.
Source_Table	The name of the physical table or a view of a table in the staging area. Note that in from/where/group by rows this column contains the text that will be passed to the database. This text may change from one database to another.
Source_Table_Ldesc	Not used by the mapping generator. It is a free format text field where you can store the description of the table for your own use if you please. The table description will be loaded into the metadata database.
Source_Column_Number	Not used by the generator. It is the number of the column in the source table.
Source_Column_Name	The name of the column in the source table.
Source_Column_Ldesc	Not used by the mapping generator. It is a free format text field where you can store the description of the column for your own use if you please. The column description will be loaded into the metadata database.
Use_flag_2	Not used by the generator. It is a place where you can record whether the source column is to be used in the data warehouse or not.
Source_Col_Data_Type	Not used by the generator. It is the data type of the source column. It is usually loaded into the spreadsheet so the architect can tell the data type of the source data being written to a column without having to refer to the source table directly.
Source_Col_Length	Not used by the generator. It is the length of the source column. It is usually loaded into the spreadsheet so the architect can tell the length of the source data being written to a column

	without having to refer to the source table directly.
Source_Col_Precision	Not used by the generator. It is the precision of the source column. It is usually loaded into the spreadsheet so the architect can tell the precision of the source data being written to a column without having to refer to the source table directly.
Source_Col_Scale	Not used by the generator. It is the scale of the source column. It is usually loaded into the spreadsheet so the architect can tell the scale of the source data being written to a column without having to refer to the source table directly.
Source_Col_Null	Not used by the generator. It indicates if the source column can be null. It is usually loaded into the spreadsheet so the architect can tell the nullability of the source data being written to a column without having to refer to the source table directly.
Source_Col_Required	Not used by the generator. It is a flag which can be filled in to say if the column is required in the data warehouse. It is similar to the previous used column.
SeETL_Table_Type	This field defines the type of the table. Valid values are: • **d1** - Type 1 dimension table • **d2** - Type 2 dimension table • **f** - Fact table • **fa** - Association fact table
Target_Owner	Owner of the target table in the data warehouse.
SeETL_View	The name of the view that will be created in both the staging area and SeETL view ddl. Note. It is recommended that this column is filled in for all columns for a SeETL view even the from, where, group by rows. This is because this column and the next column (column number) are used to sort the spreadsheet prior to running the generator.
SeETL_Column_Number	Not used by the generator. This is the number of the column in the SeETL View. It is recommended that this column is filled in for all columns for the view as this field used with the SeETL View column is used to sort the spreadsheet for preparation in running the generator.
SeETL_View_Column_Name	The name of the column to be created in the view for both the staging area and the target SeETL view.
Target_Table	The name of the physical table/view that the SeETL will write to through the view. In a 'F'rom row this column stores the sql that will be passed to the 'from' clause of the database. Note that this clause varies from database to database.
Target_Table_Exists_Ind	Not used by the generator. It is a flag to indicate to the DBA performing table maintenance whether the physical table already exists in the data warehouse.
Target_Column	The name of the physical column that the SeETL will write to via the view.
Target_Column_Exists_Ind	Not used by the generator. It is a flag to indicate to the DBA performing table maintenance whether the physical column already exists in the data warehouse.
Target_Column_Key_Ind	Not used by the generator. It is a field that indicates to the DBA performing table maintenance whether this field is a part of the key for the table.
Use_In_PV_Ind	This column defines whether the row will be exposed in a presentation view. It allows for fields to be mapped and placed into the DW but not exposed to any presentation view.
PV_Owner	Owner of the target table presentation views in the data warehouse.
PV_View	This is the name of the presentation view through which the data will be presented.
PV_Column	This is the name of the column through which the presentation view will present data.
PV_Column_Number	This is the number of the column in the presentation view. The user should note that when generating the presentation view the generator will sort the columns according to table name and this column.

	This allows for the ability to merge data from multiple tables into one presentation view. It also facilitates presenting data from one table in multiple presentation views. This also allows for the ability to sort columns and to present them in the order the user wants to. Please note that the limitation today is that the spreadsheet does not support presenting one column in one physical DW table through 2 presentation views. In this case you must write a separate view statement.
Action_Reqd	Not used by the generator. It is a field that can be used as a flag to note that further action is required for this row. For example, the column must be defined into the target data warehouse.
Notes1	Not used by the generator. It is a field to store free format text as notes about the mapping row or anything else that you would like to keep notes about.
Notes2	Not used by the generator. It is a field to store free format text as notes about the mapping row or anything else that you would like to keep notes about.
Notes3	Not used by the generator. It is a field to store free format text as notes about the mapping row or anything else that you would like to keep notes about.
User_Col_01	We have provided 10 user columns in case you have columns you want to add and load into the ctl_mapping_ss table that are not needed in any other area of the generator.
User_Col_02	We have provided 10 user columns in case you have columns you want to add and load into the ctl_mapping_ss table that are not needed in any other area of the generator.
User_Col_03	We have provided 10 user columns in case you have columns you want to add and load into the ctl_mapping_ss table that are not needed in any other area of the generator.
User_Col_04	We have provided 10 user columns in case you have columns you want to add and load into the ctl_mapping_ss table that are not needed in any other area of the generator.
User_Col_05	We have provided 10 user columns in case you have columns you want to add and load into the ctl_mapping_ss table that are not needed in any other area of the generator.
User_Col_06	We have provided 10 user columns in case you have columns you want to add and load into the ctl_mapping_ss table that are not needed in any other area of the generator.
User_Col_07	We have provided 10 user columns in case you have columns you want to add and load into the ctl_mapping_ss table that are not needed in any other area of the generator.
User_Col_08	We have provided 10 user columns in case you have columns you want to add and load into the ctl_mapping_ss table that are not needed in any other area of the generator.
User_Col_09	We have provided 10 user columns in case you have columns you want to add and load into the ctl_mapping_ss table that are not needed in any other area of the generator.
User_Col_10	We have provided 10 user columns in case you have columns you want to add and load into the ctl_mapping_ss table that are not needed in any other area of the generator.

Example DDL for the table is as follows:

```
create table dbo.ctl_mapping_ss
    (  pk_spreadsheet_name          varchar     (255)           default 'unknown'
      ,pk_spreadsheet_version       varchar     (255)           default 'unknown'
      ,pk_mapping_row               integer                     default 0
      ,Use_flag_1                   varchar     (255)           default 'unknown'
      ,Row_Type                     varchar     (255)           default 'unknown'
      ,Source_Name                  varchar     (255)           default 'unknown'
      ,Source_Owner                 varchar     (255)           default 'unknown'
      ,Source_Table                 varchar     (255)           default 'unknown'
      ,Source_Table_Ldesc           varchar     (4000)          default 'unknown'
      ,Source_Column_Number         integer                     default 0
      ,Source_Column_Name           varchar     (255)           default 'unknown'
      ,Source_Column_Ldesc          varchar     (4000)          default 'unknown'
      ,Use_flag_2                   varchar     (255)           default 'unknown'
      ,Source_Col_Data_Type         varchar     (255)           default 'unknown'
      ,Source_Col_Length            varchar     (255)           default 'unknown'
      ,Source_Col_Precision         varchar     (255)           default 'unknown'
      ,Source_Col_Scale             varchar     (255)           default 'unknown'
      ,Source_Col_Null              varchar     (255)           default 'unknown'
      ,Source_Col_Required          varchar     (255)           default 'unknown'
      ,SeETL_Table_Type             varchar     (255)           default 'unknown'
      ,Target_Owner                 varchar     (255)           default 'unknown'
      ,SeETL_View                   varchar     (255)           default 'unknown'
      ,SeETL_Column_Number          integer                     default 0
      ,SeETL_View_Column_Name       varchar     (255)           default 'unknown'
      ,Target_Table                 varchar     (255)           default 'unknown'
      ,Target_Table_Exists_Ind      varchar     (255)           default 'unknown'
      ,Target_Column                varchar     (255)           default 'unknown'
      ,Target_Column_Exists_Ind     varchar     (255)           default 'unknown'
      ,Target_Column_Key_Ind        varchar     (255)           default 'unknown'
      ,Use_In_PV_Ind                varchar     (255)           default 'unknown'
      ,PV_Owner                     varchar     (255)           default 'unknown'
      ,PV_View                      varchar     (255)           default 'unknown'
      ,PV_Column                    varchar     (255)           default 'unknown'
      ,PV_Column_Number             integer                     default 0
      ,Action_Reqd                  varchar     (255)           default 'unknown'
      ,Notes1                       varchar     (4000)          default 'unknown'
      ,Notes2                       varchar     (4000)          default 'unknown'
      ,Notes3                       varchar     (4000)          default 'unknown'
      ,User_Col_01                  varchar     (255)           default 'unknown'
      ,User_Col_02                  varchar     (255)           default 'unknown'
      ,User_Col_03                  varchar     (255)           default 'unknown'
      ,User_Col_04                  varchar     (255)           default 'unknown'
      ,User_Col_05                  varchar     (255)           default 'unknown'
      ,User_Col_06                  varchar     (255)           default 'unknown'
      ,User_Col_07                  varchar     (255)           default 'unknown'
      ,User_Col_08                  varchar     (255)           default 'unknown'
      ,User_Col_09                  varchar     (255)           default 'unknown'
      ,User_Col_10                  varchar     (255)           default 'unknown'
)
;
```

4.2.12.ctl_file_cycle_control

The table ctl_file_cycle_control is not used by SeETL [RT]. However, it is made available to increase the level of auditability of an ETL Subsystem. We have used it in many projects in the past.

The idea of the file cycle control is to create a control table which records the number of times a file has been processed and how many rows were processed each time. It is an 'increase' or 'improvement' on the audit processing. The audit processing logs the number of rows processed by each program. The audit table is not meant to be particularly queryable as it is just a log. The ctl_file_cycle_control is meant to be queryable, joinable and used as a part of a well designed audit process.

It is also possible, and we have done this on many projects, to update the file cycle control table such that the rows that are sent into the data warehouse are tagged with the batch number and the file cycle number. It is then possible to tell which batch and which file cycle a row was inserted into or last updated in the data warehouse. This has proven very valuable over the years for tracking rows and detecting the introduction of errors and correcting them.

For example, if it is known that a specific calculation error happened to be introduced on a particular date when some fixes were applied it is possible to only apply corrections to rows updated after that particular date.

There is not code to update the file cycle control tables. It is expected that the user write some small sql snippets or stored procedures to insert rows into this table.

It is recommended that careful consideration be given to using file cycle numbers if your data warehouse needs a slightly better level of auditing than just the audit logs.

The ctl_file_cycle_control table contains the following fields.

Field Name	Description
pk_view_name	This is the name of the mapping that is being used to move data from source to target. It is also the name of the view that the mapping will extract data from.
pk_file_cycle_number	This is the cycle number of the mapping. Note that a mapping may only be executed a maximum of once in any particular batch.
table_name	This is the name of the table that is affected as a target for this mapping.
batch_date	This is the batch date of the batch that processed the mapping. It is not necessarily the system date.
batch_number	This is the batch number of the batch that processed this file cycle.
number_of_records	This is the number of records that were found in the view when counted by the SQL that is generated to count input rows.

Example DDL for the table is as follows:

```
create table dbo.ctl_file_cycle_control
    (  pk_view_name                                 varchar (255)     not null
    ,  pk_file_cycle_number                         integer           not null
    ,  table_name                                   varchar (255)     not null
    ,  batch_date                                   datetime          not null
    ,  batch_number                                 integer           not null
    ,  number_of_records                            integer           not null
    ,  constraint          pk_ctl_file_cycle_control       primary key
       ( pk_view_name
       , pk_file_cycle_number
       )
    )
```

4.2.13.ctl_sequence_nums

The table ctl_sequence_nums table is used by SeETL [RT] to allocate sequence numbers to fact tables only.

The idea of sequence numbers on fact tables is this. There are many valuable and useful things that can be done by placing sequence numbers onto the front of fact tables. We have done many projects like this. However, SeETL[RT] did not originally support the generation of sequence numbers on fact tables. This support was left to the database. However, in some databases the introduction of sequence numbers at the database level is a significant performance issue. So we have added sequence numbers to SeETL [RT]. They are allocated during the attribution process and the allocation of the numbers is very fast.

The process is as follows.

If the user tells the attribution process to use sequence numbers and provides a sequence name then that name will be looked up by the attribution process to see what the last key allocated was. It will then update the last key allocated by seq_group_block size and it will give a sequence number to each row until it has exhausted all the sequence numbers in the seq_group_block. It will then return to this table to get another set of sequence numbers.

Currently this field is an integer so tables up to 4B rows are supported of negative sequence numbers are used. We will consider changing the sequence number to bigint when we have a client who has more than 4B rows in a single fact table using SeETL [RT]. That should be interesting!!

The seq_locked field has a specific purpose. In many cases a single set of input data to be processed is broken into multiple files. This is especially the case in telcos. However, we do not want sequence numbers to clash. In these cases the processing the multiple attribution processes will be going to the table to retrieve blocks of sequence numbers.

In order to make sure there is no overlap or duplication of sequence numbers each of the attribution processes can only retrieve keys when the seq_locked flag is 'N' before it starts the retrieval and update process.

Of course, this introduces contention on this table in the attribution processing. The Data Warehouse Architect must determine the most appropriate size of the seq_group_size such that:

1. Contention is minimised.
2. The gaps in the sequence numbers caused by unused numbers is not so large that the number ranges will run out of valid integers any time soon.

We expect that there will be no problems for sequence numbers for any table where the number of rows is less than 3 billion in a single table.

The ctl_sequence_nums contains the following fields.

Field Name	Description
pk_int_key	This is an integer key that makes the rows unique.
sequence_name	This is the name of the sequence that is passed to the fact table processing program AT01/02 so that it can find the correct sequence to select so as to get the correct set of numbers.
last_key_allocated	This is the last sequence number that has been allocated to be used.
seq_group_size	This is the group size of sequence numbers to allocate. For large fact tables like call detail records it is expected to set this to about 10,000 so that there is not contention for getting new sequence numbers. The attribution programs muse lock this table prior to fetching a range of sequences. If the range is too small excessive locking might take place slowing down processing. If the range is too large then gaps occur in the range and wasted numbers occur.
seq_locked	This is a flag to tell the attribution processing programs if some other program is currently getting a set of sequences. The row must be free before a program can ask for a lock and retrieve some sequence numbers.

Example DDL for the table is as follows:

```
create table dbo.ctl_sequence_nums
    (  pk_int_key             integer           not null primary key
    ,  sequence_name          varchar     (255) not null default 'unknown'
    ,  last_key_allocated     integer           not null default 0
    ,  seq_group_size         integer           not null default 0
    ,  seq_locked             varchar     (1)   not null default 'N'
    )
```

4.2.14.ctl_batch_pre_req

The ctl_batch_pre_req table is defined as follows in SQL Server 2005. It can be created in any ODBC compliant database with similar data types for each field.

The following table explains the values of each column and the functions available.

<table>
<tr><th>Column Name</th><th>Function</th></tr>
<tr><td>pk_process_batch_name</td><td>This is the batch name assigned for the specific batch. It can be any character string up to 255 characters. It is recommended that the batches are given some form of meaningful name that sorts by the batch names.

Note that the batch name is not associated with the pk_batch_number in the ctl_batch_control table. The pk_batch_number is used to record sequential numbers of batches, usually based on daily batch processing. The pk_process_batch_name is the name of a batch that will be executed on some form of scheduled basis. This means the same pk_process_batch_name may be run many times and have many pk_batch_numbers recorded against it.

Note also that Oracle treats data fields as case sensitive so the value assigned to pk_process_batch_name must be in the same case across the different tables in which it is used if you are using Oracle as the database for the scheduler.

Example: batch001.

Default: ‘unknown’</td></tr>
<tr><td>pk_pre_req_number</td><td>A batch may have many pre-requisites. All of which must be true/passed prior to the start of the batch. For example, it might be desirable to say that a batch must not start until a specific file is present and a certain time of day has been reached.

The Batch Processing Scheduling Utility does not currently support the ‘or’ operator for multiple pre-requisites.

This is the number of one or more batch pre-requisites. It is a sequential number used to sort the pre-requisites of a batch. The pre-requisites will be testing in the order they are presented. If a pre-requisite fails the later pre-requisites for the batch are not tested.

Examples: 1, 2, 3, 4 etc.

Default: 0</td></tr>
<tr><td>pre_req_type</td><td>There are a number of types of pre-requisites that can be defined for a batch. These pre-requisite types are used in conjunction with the other fields on this table to determine the exact pre-requisite for the starting of a batch.

The types and their descriptions are as follows:

<table>
<tr><th>Pre-Requisite Type</th><th>Description</th></tr>
<tr><td>FileExists</td><td>This pre-requisite row indicates that a specific file must exist prior to the batch being started. The full path name of the file to check is in field file_exists_file_name.

The file is checked using the fopen statement so the string in the field file_exists_file_name must be valid to be passed as a parameter to fopen.</td></tr>
<tr><td>Daily</td><td>This pre-requisite row indicates that the batch will be run daily at a specific time. The time for the batch to start is defined in start_hour and start_minute. When the system clock goes past the start hour every day the batch will be considered to pass the pre-requisite.</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td>Weekly</td><td>This pre-requisite row indicates that the batch will be run on a specific day of the week at a specific time. The day that the batch will start in is stored in the field day_of_week. The time for the batch to start is defined in start_hour and start_minute. When the system clock goes past the start hour on the day of week specified the batch will be considered to pass the pre-requisite.</td></tr><tr><td>Monthly</td><td>This pre-requisite row indicates that the batch will be run on a specific day of the month at a specific time. The day that the batch will start in is stored in the field day_of_month which is an integer rather than a character string. The time for the batch to start is defined in start_hour and start_minute. When the system clock goes past the start hour on the day of the month specified the batch will be considered to pass the pre-requisite. Note that if the intention is to start the batch on the last day of the month the value 99 can be used for the day of the month. Leap years are not currently supported.</td></tr><tr><td>OneTime</td><td>It is possible that you would like to run a batch only once and would like to schedule it to run at a certain time rather than have to submit it yourself. For example, during testing this would be quite handy to make sure that the batch does not accidentally get run a second time. For a ‘OneTime’ run the timestamp to start the job is stored in start_timestamp. When the system clock goes past the start_timestamp the batch is said to pass this pre-requisite.</td></tr></table>
file_exists_file_name	This is the fully qualified pathname of the file that must exist before the batch will be started when the pre-requisite type is set to ‘FileExists’.
day_of_week	This is the day of the week that the batch will be started when the pre-requisite type is set to ‘Weekly’. Valid values are: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.
day_of_month	This is the day of the month that the batch will be started when the pre-requisite type is set to ‘Monthly’. Valid values are: 1-31, and 99. Note that 99 is interpreted as ‘last day of the month’. Leap years are currently not supported.
start_hour	When the pre-requisite type is set to ‘Daily’, ‘Weekly or ‘Monthly’ this field stores the hour in which the batch will start. The scheduler uses the 24 hour clock. To start a batch in the hour after 6am this field is set to 6. To start a batch in the hour after 6pm this field is set to 18.
start_minute	When the pre-requisite type is set to ‘Daily’, ‘Weekly or ‘Monthly’ this field stores the minute of the hour at which the batch will start. To start a batch at 6:06 am you would specify start_hour=6 and start_minute=6.

	The batch will be started in the first check of the pre-requisites to occur after the system clock goes past 6:06am. To start a batch at 6:18pm you would specify start_hour=18 and start_minute=18. The batch will be started in the first check of the pre-requisites to occur after the system clock goes past 6:18pm.
start_timestamp	When the pre-requisite type is set to 'OneTime' this field contains the timestamp at which the batch should start. The batch will start in the first check of per-requisites after the system clock passes the timestamp.
short_description	This column contains a short description of the Batch that will be printed onto the reports for the batch processing. It is intended to assist the support personnel understand the batches a little more easily.
long_description	This column contains a long description of the Batch that will be printed onto the reports for the batch processing. It is intended to assist the support personnel understand the batches a little more easily.

Each row in this table represents a batch of programs that can be scheduled to be run at certain points in time or when certain files exist on the machine.

Note that pre-requisites can be combined so that it is possible to force a batch to wait until a certain point in time and a certain file exists on the system. This way it is possible to not start a batch early because a signal file has been placed onto the system. This is often the case when data is being sent to the data warehouse server by a system you have no control over but you do not want to start batch processing until after hours.

Example DDL for the table is as follows:

```
create table dbo.ctl_batch_pre_req
    (  pk_process_batch_name        varchar (255)     not null    default 'unknown'
      ,pk_pre_req_number            integer           not null    default 0
      ,pre_req_type                 varchar (020)     not null    default 'unknown'
      ,file_exists_file_name        varchar (4000)    not null    default 'unknown'
      ,day_of_week                  varchar (0020)    not null    default 'unknown'
      ,day_of_month                 integer           not null    default 0
      ,start_hour                   integer           not null    default 0
      ,start_minute                 integer           not null    default 0
      ,start_timestamp              datetime
      ,short_description            varchar (4000)    not null    default 'unknown'
      ,long_description             varchar (4000)    not null    default 'unknown'

    )
;
```

4.2.15.ctl_batch_run_log

Batches are monitored using the batch run log table.

Column Name	Function
pk_batch_number	This is the number of the batch from the ctl_batch_control table. It identifies the 'batch' that this particular batch was run under. It is important to understand that there are two meanings for a batch. The first meaning is that a batch is a set of processes that can be run. The second meaning is that a batch is the instance of that set of processes actually being run. When processing is performed on a repeated basis the scheduler checks to see if the repetitive processing has already been run in the current 'batch'. If it has been run successfully it will not be started again. This way the daily processing of the batch cannot be accidentally re-run.
pk_process_batch_name	This is the batch name assigned for the specific batch. It is the same batch name as occurs on the ctl_batch_pre_req table. A Process Group is placed into a batch by using the same name that defines the batch. Example: batch001. Default: 'unknown'
pk_started_tstamp	This is the start timestamp of the batch.
ended_tstamp	This is the end timestamp of the batch.
end_status	This is the end status of the batch. If can be 'Running', 'Successful' or 'Abended'. Once a batch is Abended it can only be re-started by the user going into this table and changing the end_status from 'Abended' to 'Restart'. When the end_status is set to 'Restart' the scheduler will re-submit the Process Groups that make up the batch and the Process Groups will restart at the point of the abend.

Example DDL for the table is as follows:

```
create table dbo.ctl_batch_run_log
    (  pk_batch_number        integer           not null    default 0
      ,pk_process_batch_name  varchar     (255) not null    default 'unknown'
      ,pk_started_tstamp      datetime          not null
      ,ended_tstamp           datetime
      ,end_status             varchar     (20)  not null    default 'unknown'
    )
;
```

4.2.16.ctl_column_defs

The Metadata Loading Utility loads table metadata into a table so that it can be used in reports or analysis for mapping jobs. The table name can be passed as a parameter. Also, the column names are NOT used inside the program however the fields are expected to be of the following format and the first 4 columns are expected to be the primary key. It is probably advisable to use the following table definition but it is not 'required'.

Please note that the Metadata Loading Utility actually issues unconditional delete statements to remove a row from the target table before issuing the insert. This is to make it easily re-runnable and it was easier to code than to test for the violated constraint. Therefore the first 4 columns must start with "pk_" even if you choose to call them something else.

Column Name	Function
pk_catalog_name	The name of the database or 'catalog' when viewed as an ODBC data source.
pk_schema_name	The owner or schema name from the database.
pk_table_name	The name of the table/view for the column.
pk_column_name	The name of the column.
column_number	The ordinal position of the column in the table/view.
primary_key_ind	An indicator as to whether the column is a primary key. Set to 1 for a field that is a part of the primary key.
data_type	This is the numeric field that is the ODBC return value for a data type. It is decoded by the ctl_data_types table.
column_length	This is the length of the column.
decimal_digits	This is the number of decimal digits for the column.
nullable	This is a flag to indicate if the field is nullable or not. It is set to 1 if the field is nullable.

Example DDL for the table is as follows:

```
create table dbo.ctl_column_defs
    (  pk_catalog_name        varchar     (255)       not null    default 'unknown'
      ,pk_schema_name         varchar     (255)       not null    default 'unknown'
      ,pk_table_name          varchar     (255)       not null    default 'unknown'
      ,pk_column_name         varchar     (255)       not null    default 'unknown'
      ,column_number          integer                 not null    default 0
      ,primary_key_ind        integer                 not null    default 0
      ,column_length          integer                 not null    default 0
      ,decimal_digits         integer                 not null    default 0
      ,nullable               integer                 not null    default 0
   )
;
```

4.2.17.ctl_commands

So, up until now you have not read anything about actually executing Processes (commands) in the scheduler. It has all been about Batches and Process Groups. Of course, at some point in the documentation of the Scheduler we must actually document how to run a Process (command).

The following table explains the values of each column and the functions available.

Column Name	Function
pk_process_batch_name	This is the batch name assigned for the specific batch. It is the same batch name as occurs on the ctl_batch_pre_req table. A Process Group is placed into a batch by using the same name that defines the batch. Example: batch001. Default: 'unknown'
pk_process_group_name	This is the name of the Process Group. The name of the process group links this Process to the Process Group. Example: group001, group002, group003 etc.
pk_process_step_number	This is the number of the Process in the Process Group. Processes are executed sequentially in the Process Group and the return code from each of the processes must be zero for the Process Group to continue processing. Zero happens to be the standard return code of SeETL [RT].
process_command_name	This is a character name that the user can give to a command. This name can be more descriptive than the program name itself. For example CTLU001.exe is the Data Transfer Utility. So the user might want to put DTU in the process_command_name as a reminder as to what process is actually running.
process_program_name	This is the program name that will be submitted to the 'system' command inside windows/unix. Previous versions required that this command be visible in the path of the userid running SeETL [RT]. Typically the command was in a file visible to the operating system and $PATH. In version 2.1 this approach can still be used. However, we have introduced a new table called ctl_process_commands. This table contains commands that can also be submitted to the operating system. The table can be used to eliminate the need to place SeETL [RT] commands into files visible to the $PATH. See below for a full description of the ctl_process_commands table. The program name can actually be any program name that can be submitted at the command line of windows/unix. Thus the Batch Processing Scheduling Utility can be used to schedule and run any commands on windows and unix. (A handy utility indeed!) Example. To run the Data Transfer Utility on windows this field would contain CTLU001.exe.
process_program_parms	This field contains the parameters to pass to the program. In the case of SeETL [RT] this is the set of parameters needed to run whichever program that is being submitted using the process_program_name. We have not included an example here as there are plenty of other examples in this document.
process_program_name_fail	If the command in process_program_name fails this is the program name that will be submitted to the 'system' command inside windows/unix to perform some kind of 'failed' command processing. The program name must be found in the path of the operating system. The scheduler does not allow for you to specify the path of the executable program.
process_program_parms_fail	This field contains the parameters to pass to the program that is run on the failure of the program specified in process_program_name.
short_description	This column contains a short description of the command that will be printed onto

	the reports for the batch processing. It is intended to assist the support personnel understand the batches a little more easily.
long_description	This column contains a long description of the command that will be printed onto the reports for the batch processing. It is intended to assist the support personnel understand the batches a little more easily.
cmd_pre_req_flag	This flag is set to 'Y' when the command has a pre-requisite file associated with it. If this flag is not set to 'Y' then the remainder of the line is ignored.
cmd_pre_req_type	This field is set to 'FileExists' or 'FileNotExists' depending on whether the user wants to test for the existence or non-existence of a file to determine if the command should be executed.
cmd_file_exists_file_name	This is the name of the file that will be tested for existence or non-existence when this pre-requisite is set.

So, in this brief introduction you can see that a Process (command) is part of a Process Group and the Process Group is part of a Batch. This is how the hierarchy of the processes are constructed.

This is the commands table.

```
create table dbo.ctl_commands
    (  pk_process_batch_name        varchar (255)     not null    default 'unknown'
      ,pk_process_group_name        varchar (255)     not null    default 'unknown'
      ,pk_process_step_number       integer           not null    default 0
      ,process_command_name         varchar (255)     not null    default 'unknown'
      ,process_program_name         varchar (255)     not null    default 'unknown'
      ,process_program_parms        varchar (4000)    not null    default 'unknown'
      ,process_program_name_fail    varchar (255)     not null    default 'unknown'
      ,process_program_parms_fail   varchar (4000)    not null    default 'unknown'
      ,short_description            varchar (4000)    not null    default 'unknown'
      ,long_description             varchar (4000)    not null    default 'unknown'

/* These three fields added to enable conditional running of commands.... */

      ,cmd_pre_req_flag             varchar (1)       not null    default 'N'
      ,cmd_pre_req_type             varchar (50)      not null    default 'unknown'
      ,cmd_file_exists_file_name    varchar (4000)    not null    default 'unknown'
)
;
```

4.2.18.ctl_datastage_parms

This table maintains the parameters for the DataStage jobs. It can be a view or a table. If you would like to be able to customise the parameters according to the specific jobs that are run you would add a column such as job_name or job_group and you would present a different view for the different jobs or job groups.

The program expects to see three columns as below for its parameters and this is what the view must present.

Column Name	Function
pk_parm_number	This is an integer to create a primary key for the row.
parm_name	This is the name of the parameter that will be fetched and sent to the DataStage job.
parm_value	This is the value of the parameter

Example DDL for the table is as follows:

```
create table dbo.ctl_datastage_parms
    (  pk_parm_number                    integer    not null  primary key
    ,  parm_name                         varchar (255)
    ,  parm_value                        varchar (4000)
    )
;
```

4.2.19.SeETL DDL Generator

In early versions of the SeETL we created the ability to create tables on both SQL Server and Oracle. This was quite useful until the databases added more and more features that were more and more difficult to support. In the end the utility fell into dis-use. The latest release includes a new sheet in the workbook to create tables that is more flexible. It is expected that this old version of the utility to generate DDL will simply be “end of lifed” at some point in time. It is included here for completeness as it is still available and still supported.

Is there anything more tedious than trying to maintain SQL DDL by hand? Perhaps only trying to maintain a lot of views of tables using one of the ‘productivity tools’ for database design. And who has never had the problem of getting the wrong number of columns in the create view statement when compared to the select statement supporting the view?

The SQL Server 2005 DDL Generator Utility reads a simple model stored in two tables in the target database and from them generates the DDL defined by the data in the tables. It’s not meant as a replacement for ‘repository’ products. It is meant as a quick and easy way of generating DDL for portions of the data warehouse. Especially making sure that the views line up etc.

4.2.19.1. ctl_ddl_gen_table

The ctl_ddl_gen_table defines the tables/views that will be generated for this run of the DDL generator. Each record in ctl_ddl_gen_table correlates to one table and one view being generated. The DDL generator always produces the table DDL and the view DDL and places the output into separate files so that you can inspect the DDL prior to running it against the database.

Field Name	Description
process_row	This has the values ‘y’ or ‘n’ depending on whether you want the DDL generator to generate the DDL for this table on this run of the DDL generator.
Table_name	This is the name of the table that will be generated by the DDL generator.
View_name	This is the name of the view that will be generated by the DDL generator.
Table_type_ind	This field defines the type of table that will be generated. We recommend you fill in the column as follows: ‘d’ – Dimension table ‘f’ – Fact table ‘c’ – Control table ‘s’ – Staging table ‘e’ – Extract table ‘w’ – Work table Note that this field is used by the DDL generator to place the where level_col = ‘detail’ clause on the tables defined as ‘d’imension tables.
Table_file_group	This is the SQL Server file group that you would like the physical table to be placed in. You can place the primary index and the data in different file groups if you wish. If you want to partition your fact tables across different file groups you will need to generate the DDL using your usual methods.
index_file_group	This is the SQL Server file group that you would like the primary index to be placed in. You can place the primary index and the data in different file groups if you wish.
Table_ddl_file	This is the fully qualified name of the file to which you would like to write the DDL for the table.
View_ddl_file	This is the fully qualified name of the file to which you would like to write the DDL for the view.

Example DDL for the table is as follows:

```
create table dbo.ctl_ddl_gen_table
    (  process_row            varchar     (001)   not null default 'n'
    ,  table_name             varchar     (255)   not null default 'unknown'
    ,  view_name              varchar     (255)   not null default 'unknown'
    ,  table_type_ind         varchar     (001)   not null default 'U'
    ,  table_file_group       varchar     (030)   not null default 'unknown'
    ,  index_file_group       varchar     (030)   not null default 'unknown'
    ,  table_ddl_file         varchar     (255)   not null default 'unknown'
    ,  view_ddl_file          varchar     (255)   not null default 'unknown'
    )
;
```

Some example values for this table are:

Proc Row?	Table	View Name	Type	Table File Group	Index File Group	Table DDL	View DDL
y	f_order_fact	f_order_fact_view	f	ctlfg01	ctlfg01	Filename	Filename
y	f_order_fact_summ	f_order_fact_summ_view	f	ctlfg01	ctlfg01	Filename	Filename

4.2.19.2. ctl_ddl_gen_columns

The ctl_ddl_gen_columns table defines the columns within each of the tables. Each record in ctl_ddl_gen_columns correlates to column in a table and view.

Field Name	Description
table_name	This is the same table name as specified in ctl_ddl_gen_table. This record is looked up by the DDL generator after it has identified the table as one of the tables for which to generate DDL.
table_column_name	This is the name of the column in the table. It will be used in both the create table statement and the create view statement that is generated.
view_column_name	This is the name of the column in the view defined to be created in ctl_ddl_gen_table. This name is only used in the generation of the DDL for the view.
db_data_type	This is the data type of the column that will be passed to the database in the DDL. It must be a valid data type in value DDL format for the database manager to process. For example, 'integer', varchar (10) and money are all valid data types for SQL Server.
col_number_in_table	This is the number of the column within the table. Strictly speaking it is not necessary in the relational environment. However, it is always useful to have columns ordered as you would like to see them. This column can just be an ascending number to place the columns in the table in the order that you would like them. The DDL generator sorts the columns into this order to produce the create table and create view DDL.
col_number_in_pk	This is the position of the column in the primary key. If a column is not a part of the primary key the column should be set to 0. Note that the DDL generator is expecting that every table has a primary key. If you want to use the DDL generator to generate the DDL for tables without primary keys you will have to edit the DDL after it is produced. The main reason for this is that it is a rare case indeed in a data warehouse that a table does not have a primary key. All dimension tables have the level column and dim char ky fld as primary keys and all summary fact tables have their combination of integer keys as primary keys. It is recommended that detail level fact tables have primary keys to provide a level of protection against double processing of a file into a fact table.
flag_column_nullable	This column is set to 'y' if the column is nullable and 'n' if the column is to be defined as not null.

Example DDL for the table is as follows:

```
create table dbo.ctl_ddl_gen_columns
    (  table_name             varchar     (030)   not null default 'unknown'
    ,  table_column_name      varchar     (030)   not null default 'unknown'
    ,  view_column_name       varchar     (030)   not null default 'unknown'
    ,  db_data_type           varchar     (030)   not null default 'unknown'
    ,  col_number_in_table    integer             not null default 0
    ,  col_number_in_pk       integer             not null default 0
    ,  flag_column_nullable   varchar     (001)   not null default 'N'
    )
;
```

Some example values for this table are:

Table Name	Table Column Name	View Column Name	DB Data Type	Col Number in Table	Col Number in PK	Null?
f_order_fact	pk_time_key	pk_time_key	integer	1	1	n
f_order_fact	pk_prod_key	pk_prod_key	integer	2	2	n
f_order_fact	pk_cust_key	pk_cust_key	integer	3	3	n
f_order_fact	pk_vendor_key	pk_vendor_key	integer	4	4	n
f_order_fact	invoice_qty	invoice_qty	integer	5	0	n
f_order_fact	invoice_amt	invoice_amt	money	6	0	n
f_order_fact	discount_amt	discount_amt	money	7	0	n
f_order_fact	product_code	product_code	varchar (6)	8	0	n
f_order_fact	customer_code	customer_code	varchar (6)	9	0	n
f_order_fact	vendor_code	vendor_code	varchar (6)	10	0	n
f_order_fact	invoice_num	invoice_num	varchar (6)	11	0	n
f_order_fact	invoice_remark	invoice_remark	varchar (30)	12	0	n
f_order_fact	order_date	order_date	datetime	13	0	n
f_order_fact	payment_date	payment_date	datetime	14	0	n
f_order_fact	delivery_date	delivery_date	datetime	15	0	n
f_order_fact_summ	pk_time_key	pk_time_key	integer	1	1	n
f_order_fact_summ	pk_prod_key	pk_prod_key	integer	2	2	n
f_order_fact_summ	pk_cust_key	pk_cust_key	integer	3	3	n
f_order_fact_summ	pk_vendor_key	pk_vendor_key	integer	4	4	n
f_order_fact_summ	invoice_qty	invoice_qty	integer	5	0	n
f_order_fact_summ	invoice_amt	invoice_amt	money	6	0	n
f_order_fact_summ	discount_amt	discount_amt	money	7	0	n

4.2.20.ctl_proc_grp_pre_req

Setting up batches does not actually allow you enough control to make full use of parallel processing because it is not possible to run multiple batches at a time and to ensure that multiple submitted batches run to conclusion before starting another batch.

To enable such control of parallel processing the Batch Processing Scheduling Utility has implemented the concept of:

- Process groups and
- Processes (also called commands).

The following table is the Process Group Pre-Requisite definition. Most importantly, a Process Group can have another Process Group as a pre-requisite. This allows the user to build semaphores out of process groups so that many process groups can be dependent on one process group.

The following table explains the values of each column and the functions available.

Column Name	Function
pk_process_batch_name	This is the batch name assigned for the specific batch. It is the same batch name as occurs on the ctl_batch_pre_req table. A Process Group is placed into a batch by using the same name that defines the batch. Example: batch001. Default: 'unknown'
pk_process_group_name	This is the name of the Process Group. A Process Group, as the name implies, groups together a set of Processes (or commands) that will be run sequentially. That is, within the one process group, each Process must run one after the other and each process must finish with a zero return code. Any process that finishes with a return code other than zero will be considered to be abnormally ended (abended). When a Process abends the Process Group and the Batch will be marked as having been abended and the user must 'Restart' the batch in order to complete it. Just re-submitting the batch will not re-start the batch. If the user just re-submits the batch the scheduler will issue a message to say the user re-submitted an abended batch and a restart is required. This is to ensure that some user intervention takes place after a failed Process. Process group names must be unique within a specific batch. Example: group001, group002, group003 etc.
pk_pre_req_number	This is the number of the pre-requisite for the process group. A process group can have as many pre-requisites as the user likes. The pre-requisites will be tested in the order of the pre-requisite numbers. The later pre-requisites will only be tested when the earlier pre-requisites are passed.
pre_req_type	This field defines the type of pre-requisite for the process group. A Process Group may only have 2 types of pre-requisites. ProcessGroup and FileExists: FileExists is a little redundant because it is available at the batch level, however, it seems like a useful pre-requisite type to have for a process group. By making a number of Process Groups dependent on another Process Group and then making another subsequent Process Group dependent on the group of Process Groups you can construct parallel processing batches. Valid Values:

	• ProcessGroup • FileExists
file_exists_file_name	When pre_req_type is set to 'FileExists' This is the fully qualified path name of the file that must exist before the Process Group will be started.
pre_req_process_group	When pre_req_type is set to 'ProcessGroup' This is the name of the Process Group that must be completed successfully before the Process Group will be started.
short_description	This column contains a short description of the Process Group that will be printed onto the reports for the batch processing. It is intended to assist the support personnel understand the batches a little more easily.
long_description	This column contains a long description of the Process Group that will be printed onto the reports for the batch processing. It is intended to assist the support personnel understand the batches a little more easily.

Example DDL for the table is as follows:

```
create table dbo.ctl_proc_grp_pre_req
    (  pk_process_batch_name        varchar (255)     not null    default 'unknown'
      ,pk_process_group_name        varchar (255)     not null    default 'unknown'
      ,pk_pre_req_number            integer           not null    default 0
      ,pre_req_type                 varchar (020)     not null    default 'unknown'
      ,file_exists_file_name        varchar(4000)     not null    default 'unknown'
      ,pre_req_process_group        varchar (255)     not null    default 'unknown'
      ,short_description            varchar (4000)    not null    default 'unknown'
      ,long_description             varchar (4000)    not null    default 'unknown'
)
;
```

4.2.21.ctl_proc_grp_run_log

Process Groups are monitored using the process group run log table.

Column Name	Function
pk_batch_number	This is the number of the batch from the ctl_batch_control table. It identifies the 'batch' that this particular batch was run under. It is important to understand that there are two meanings for a batch.
pk_process_batch_name	This is the batch name assigned for the specific batch. It is the same batch name as occurs on the ctl_batch_pre_req table. A Process Group is placed into a batch by using the same name that defines the batch. Example: batch001. Default: 'unknown'
pk_process_group_name	This is the name of the Process Group whose activity is being logged by this record.
started_tstamp	This is the start timestamp of the Process Group. Note. The pk_ has been dropped from the beginning of this field name in Release 2.1.
ended_tstamp	This is the end timestamp of the Process Group.
end_status	This is the end status of the Process Group. If can be 'Running', 'Successful' or 'Abended'. The user should never update this field.

Example DDL for the table is as follows:

```
create table dbo.ctl_proc_grp_run_log
    (  pk_batch_number              integer           not null    default 0
      ,pk_process_batch_name        varchar (255)     not null    default 'unknown'
      ,pk_process_group_name        varchar (255)     not null    default 'unknown'
      ,started_tstamp               datetime
      ,ended_tstamp                 datetime
      ,end_status                   varchar (20)      not null    default 'unknown'
       parent_process_id            integer           not null    default 0
       child_process_id       integer           not null    default 0     )
;
```

4.2.22.ctl_proc_run_log

Processes are monitored using the process run log table.

Column Name	Function
pk_batch_number	This is the pk_batch_number from the ctl_commands table.
pk_process_batch_name	This is the pk_process_batch_name from the ctl_commands table.
pk_process_group_name	This is the pk_process_group_name from the ctl_commands table.
pk_process_step_number	This is the pk_process_step_number from the ctl_commands table.
process_command_name	This is the process_command_name from the ctl_commands table.
process_program_name	This is the process_program_name from the ctl_commands table.
process_program_parms	This is the process_program_parms from the ctl_commands table.
pk_started_tstamp	This is the start timestamp of the Process Group.
ended_tstamp	This is the end timestamp of the Process Group.
end_status	This is the end status of the Process Group. If can be 'Running', 'Successful' or 'Abended'. The user should never update this field.

Example DDL for the table is as follows:

```
create table dbo.ctl_proc_run_log
    (  pk_batch_number              integer           not null    default 0
      ,pk_process_batch_name        varchar (255)     not null    default 'unknown'
      ,pk_process_group_name        varchar (255)     not null    default 'unknown'
      ,pk_process_step_number       integer           not null    default 0
      ,process_command_name         varchar (255)     not null    default 'unknown'
      ,process_program_name         varchar (255)     not null    default 'unknown'
      ,process_program_parms        varchar (4000)    not null    default 'unknown'
      ,started_tstamp               datetime
      ,ended_tstamp                 datetime
      ,end_status                   varchar (20)      not null    default 'unknown'
       parent_process_id            integer           not null    default 0
       child_process_id       integer           not null    default 0     )
    )
;
```

4.2.23.ctl_process_commands

In Release 2.1 we introduced the ctl_process_commands table. Up until Release 2.1 the user had to place the commands that would be executed into files on the operating system and then make those commands visible in the path of the userid that would execute the Batch.

As a part of our move to place as much information as possible into the 'Mapping Spreadsheet' we have enhanced the Scheduler to optionally read the commands to be executed from the ctl_process_commands table.

When the Scheduler attempts to run a command defined in the ctl_commands table (above) it will first read this ctl_process_commands table to determine if the command exists in this table. It uses the process_program_name from the ctl_commands table to search the process_name in the ctl_process_commands table. If these two fields are the same the process_command in the ctl_process_commands table will be submitted to the operating system.

If the command does not exist in the ctl_process_commands table then the command and the parameters will be passed to the operating system unchanged.

If the command does exist in the ctl_process_commands table then the Scheduler will

1. Read the command from the ctl_process_commands table.
2. Perform the defined parameter substitution in the command.
3. Send the finished command to the operating system for execution.

In the SeETL [RT] Batch Processing Utility the parameter character is defined to be '?' and not '%' or '$' as is used in windows and unix based systems.

The reason is that it is also possible that the user will want to be able to pass windows or unix based parameters as well as SeETL [RT] based parameters to a command.

The maximum number of parameters that can be passed to a command is 9 (nine). They are specified as ?1. ?2. ?3 through to ?9. We have not allowed the use of ?0 as we felt that would be just too confusing.

The one parameter can be used in the command in many places. For example ?1 might be used in two or three places in the command itself.

There is currently very little 'error checking' around the use of parameters so we recommend that users are careful to ensure that the commands and parameters are correct before placing them into the Mapping Spreadsheet. For example ?A will not cause any substitution to take place. A substitution will only take place for ?1-?9.

The following table explains the values of each column and the functions available.

Column Name	Function
pk_int_key	This is simply a sequential number which can be used as a primary key if desired.
process_name	This is the name of the Process that is entered into the ctl_commands table. For example this command might be 'RunType1Dimension' or 'RunFactTable1' or some such other command that is short hand for the longer commands that have typically been used.

process_command	This is the full command including parameter place holders that have been placed in files in previous versions.
	
	The typical example is 'RunType1Dimension'.
	
	In this field the user could place:
	
	`CTLDM01.exe DBConnectionInParameter=DSN=SEETL3000`
	`InCatalogName=SEETL3000`
	`InSchemaName=dbo`
	`InTableName=?1 DBConnectionOutParameter=DSN=SEETL3000`
	`OutCatalogName= SEETL3000`
	`OutSchemaName=dbo`
	`OutTableName=?1`
	`ErrorMessageOutput=TargetDatabase`
	`DebugLevel=0`
	`Audit=Yes`
	`InputTableOrFileCanBeEmpty=Yes`
	
	The user could then place into the ctl_commands table a call such as
	process_program_name = RunType1Dimension
	process_program_parms = customer_dim
	
	This would then retrieve the process_command described above and substitute the ?1 parameter place holders with 'customer_dim' and then submit the final command directly to the operating system.

short_description	This column contains a short description of the process command that will be printed onto the reports for the batch processing. It is intended to assist the support personnel understand the batches a little more easily.
long_description	This column contains a long description of the process command that will be printed onto the reports for the batch processing. It is intended to assist the support personnel understand the batches a little more easily.

This data can be entered directly into this table or, more typically, entered into the SeETL [DT] Spreadsheet' to be loaded into the table using SeETL [DT].

Example DDL for the table is as follows:

```
create table dbo.ctl_process_commands
    (  pk_int_key             integer                  not null primary key
    ,  process_name           varchar        (256)     not null default 'unknown'
    ,  process_command        varchar        (4000)    not null default 'unknown'
      ,short_description      varchar        (4000)    not null default 'unknown'
      ,long_description       varchar        (4000)    not null default 'unknown'
    )
;
```

4.2.24.ctl_data_types

The various databases have data types that are slightly different. These data types are sent back to the SeETL C++ Runtime Engine as numeric values representing the more generic ODBC data type. The SeETL C++ Runtime Engine is only aware of the ODBC Data Type and not the underlying database type. To understand exactly what it is dealing with the SeETL C++ Runtime Engine is also told the database type that it is talking to via a parameter. There is a lot of code inside the SeETL C++ Runtime Engine to ensure that the engine knows exactly what it is dealing with.

In order to be able to produce reports that are somewhat understandable as to the exact data type the engine is dealing with this data types table is required to perform that translation.

Column Name	Function
pk_data_type	This is the integer returned by ODBC to tell the engine the data type that it is dealing with.
data_type_sdesc	This is user entered text that that is printed onto a report to tell the user what the underlying data type of the field is. It uses the language or specific type of the database that is being employed by the client.

Example DDL for the table is as follows:

```
create table dbo.ctl_data_types
    (  pk_data_type           integer                not null primary key
    ,  data_type_sdesc        varchar     (256)      not null default 'unknown'
    )
;
```

4.2.25.ctl_src_file_defs

The Source File Definitions tab is used to document the source files that are under consideration to be mapped into the ODS or Data Warehouse. It provides a way of managing the different stages source files might go through in being mapped into the data warehouse from original exploration to in production.

Column Name	Function
pk_int_key	This is an integer key that is simply used to be able to retrieve specific rows in queries. It has no other value or meaning.
src_table_name	If it is a table the source full table name
src_table_suffix	If it is a table the source prefix (schema)
src_table_suffix	If it is a table the source table name without prefix
file_name	If it is a file the name of the actual file
file_status	This is the status of the file or table in the process of putting it into the target data warehouse. It is recommended that you select statuses that are ascending so that the spreadsheets that can be downloaded from the SeETL Reports can be 'drilled' on Status with a customer condition of Status > 'desired status'. It is also recommended that a status be included for files that do not make it into the scope of the project as they may make it into the scope of later projects. Proposed status might be: • 00 – Discovery. • 01 – Evaluation • 03 – Out of Scope • 04 – In Scope • 05 – Staging Table defined • 06 – ODS table defined • 07 – EDW table defined • 08 – SRC to STG ETL complete • 09 – STG to ODS ETL complete • 10 – ODS to EDW ETL complete • 11 – Integration testing • 12 – System testing • 13 – In production
source_system	The name of the source system from which the file/table comes.
src_cust_org	If the file is coming from an external customer/organisation then the name of the customer/organisation.
file_desc_name	The name of the file that is used in discussions or documentation. For example, 'Sell Through from Tesco'
description	Enter a description of the file.
contact	The name of the person who is the contact for sending/creating the file/table.
our_data_own	Name and Role of the person in our company who is responsible for the relationship with the source organisation for the provision of the file. This can be NA (not applicable) if it is from Internal Systems.
our_tech_sup	The name and role of our IT person who is responsible for the timely and effective processing of this file.
granularity	Some statement of the level of granularity of the data that is in the file. This is for descriptive purposes only. • SKU? • Store? • Reseller? • Account? • CDR? • Transaction • Combination of the above?
frequency	Is the file: • Near Real time? • Hourly? • Daily?

	• Weekly? • Monthly? • Ad-hoc? • Any other?
approx_vol	Is it 1MB, 10MB, 100MB, 1GB, 10GB or more?
transfer_mech	Is the file sent through: • EDI? • Ftp? • Email?
file_format	Is the file: • Excel? • Delimited text? • Fixed format? • EDI interchange format? • XML?
no_of_files	Number of files that are frequently transferred. In some cases a 'file' is actually a collection of files that must be processed together and if that number is more than one it should be recorded here.
sys_loaded_today	If the data is loaded to a system today then the current system where the information is loaded to.
reviewed_by	The name of the person who reviewed this file to determine if it was really required in the EDW.
required_by	The name of the person who said that the file should be in scope for the EDW. Note that though a person says they believe a file should be in scope for the EDW that file should still be reviewed and checked to determine that: 1. The data is not already loaded into the EDW. 2. There is value in loading the data into the EDW.
notes1	There are five cells for notes for each file that can be completed at the Architect's discretion.
notes2	There are five cells for notes for each file that can be completed at the Architect's discretion.
notes3	There are five cells for notes for each file that can be completed at the Architect's discretion.
notes4	There are five cells for notes for each file that can be completed at the Architect's discretion.
notes5	There are five cells for notes for each file that can be completed at the Architect's discretion.

Example DDL for the table is as follows:

```
create table dbo.ctl_src_file_defs (
       pk_int_key             integer                 not null primary key
      ,src_table_name         varchar (255)           not null default 'unknown'
      ,src_table_prefix       varchar (255)           not null default 'unknown'
      ,src_table_suffix       varchar (255)           not null default 'unknown'
      ,file_name              varchar (255)           not null default 'unknown'
      ,file_status            varchar (255)           not null default 'unknown'
      ,source_system          varchar (255)           not null default 'unknown'
      ,src_cust_org           varchar (255)           not null default 'unknown'
      ,file_desc_name         varchar (255)           not null default 'unknown'
      ,description            varchar (4000)          not null default 'unknown'
      ,contact                varchar (255)           not null default 'unknown'
      ,our_data_own           varchar (255)           not null default 'unknown'
      ,our_tech_sup           varchar (255)           not null default 'unknown'
      ,granularity            varchar (255)           not null default 'unknown'
      ,frequency              varchar (255)           not null default 'unknown'
      ,approx_vol             varchar (255)           not null default 'unknown'
      ,transfer_mech          varchar (255)           not null default 'unknown'
      ,file_format            varchar (255)           not null default 'unknown'
      ,no_of_files            int                     not null default 0
      ,sys_loaded_today       varchar(255)            not null default 'unknown'
      ,reviewed_by            varchar(255)            not null default 'unknown'
      ,required_by            varchar(255)            not null default 'unknown'
      ,notes1                 varchar(4000)           not null default 'unknown'
      ,notes2                 varchar(4000)           not null default 'unknown'
      ,notes3                 varchar(4000)           not null default 'unknown'
      ,notes4                 varchar(4000)           not null default 'unknown'
      ,notes5                 varchar(4000)           not null default 'unknown'
)
```

4.2.26.ctl_mapping_doc_infa

The Mappings Documentation Infa tab is used to store information specific to generating the Informatica Programmer Specifications documentation that a client using SeETL and Informatica might like to generate.

Because SeETL is free when used as a prototyping tool we have a number of clients who use SeETL to build their prototype EDW and when they are happy with it they use Informatica to write the 'real ETL' as they have made a long term commitment to building all production ETL in Informatica.

Some of these clients have asked us to assist in the process of converting the SeETL workbook to Informatica Programmer Program Specifications. Therefore, this sheet contains all the 'extra' information required to produce such Program Specifications.

The Mappings Documentation Infa tab contains the following fields.

4.2.26.1. ctl_mapping_doc_infa_01

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_int_key	The PK Int Key is a sequence number used as the primary key to the table.
mapping_name	This is the name of the Informatica mapping to be produced.
mapping_dependencies	This lists the dependencies of this mapping. That is, this lists the mappings that must be run prior to this mapping running or, if there are many, the groups of mappings that must be run before this mapping.
view_name	The view name provided in this cell provides the 'link' to the Mappings01 workbook. The view name is used to look into the table storing the Mappings01 workbook and select out the rows that are mapped using this view name.
target_table_name	This is the name of the target table for the mapping. It is generally expected that a mapping has one target table, and when it has more than one target table it is expected there is a 'main' target table.
description	This is a description of the target table.
development_order	This is a field which can be used to document the development order of mappings. It can be used to sort results of queries against the metadata database so that the Architect can define the development order and then see progress against that order.
development_priority	This is a field where the development priority of a mapping can be set.
development_status	This is a field where the development status of a mapping can be set. It is recommended that the description used be prefixed with numerics so that it can be queried using custom in the Excel spreadsheet reports provided with SeETL. It is recommended that some sequence of statuses be developed so that the reporting spreadsheet can be drilled by Status. A suggestion is: • 00 – Unallocated • 01 – Allocated to Programmer • 02 – In Development • 03 – Unit Testing • 04 – Walkthrough Passed • 05 – Integration Testing • 06 – System Testing • 07 – In Production
developer_assigned	This is the name of the developer to whom the development of this mapping has been assigned.
estimated_hours	This is the estimate number of hours to complete this mapping.
audit_controls_reqd	This field defines whether audit controls are required in this mapping. There is a separate report that documents Audit Fields and it uses this cell to constrain that report. If this field is 'Yes' then Audit controls are required.

audit_fields_reqd	If the prior cell is set to 'Yes' then this field lists the fields/columns on the file/table that require auditing in the Informatica ETL processing.
refresh_methodology	This field documents the refresh strategy used by this mapping. In particular, is the target table to be 'insert/update' or 'truncate/load'.
rec_sel_criteria	This is an English Language description of how the developer should select rows from the source file/table.
condition_in_sq	This is the encoding that will be required in the Informatica SQ to select the source records/files from the input data.
src_table_names	This field contains the names of the tables to be read as input for this mapping.
primary_key	This field contains the names of the fields which constitute the unique key that will be used to determine whether the row in the target table should be 'insert' or 'update'. It is not the primary key on the target table which is very likely a sequence.
parameter01	These 5 fields contain documentation of parameter values that may be set in the mapping.
parameter02	These 5 fields contain documentation of parameter values that may be set in the mapping.
parameter03	These 5 fields contain documentation of parameter values that may be set in the mapping.
parameter04	These 5 fields contain documentation of parameter values that may be set in the mapping.
parameter05	These 5 fields contain documentation of parameter values that may be set in the mapping.
variable01	These 5 fields contain documentation of variable values that may be set in the mapping.
variable02	These 5 fields contain documentation of variable values that may be set in the mapping.
variable03	These 5 fields contain documentation of variable values that may be set in the mapping.
variable04	These 5 fields contain documentation of variable values that may be set in the mapping.
variable05	These 5 fields contain documentation of variable values that may be set in the mapping.
source_connection	This field contains the definition of the source connection that should be used by the mapping to connect to its input data.
target_connection	This field contains the definition of the target connection that should be used by the mapping to connect to its output tables.
err_log_db_connection	This field contains the definition of the target connection that should be used by the mapping to connect to its Error Log database.
default_fields	If some fields are set to defaults then this cell documents the names and values of those defaults.
session_names	This field documents the session name to be used for this mapping.
workflow_names	This field documents the workflow name which will run this mapping.

Example DDL for the table is as follows:

```
create table dbo.ctl_mapping_doc_infa_01 (
        pk_spreadsheet_name       varchar (255)             default 'unknown'
       ,pk_spreadsheet_version    varchar (255)             default 'unknown'
       ,pk_int_key                integer                   not null default 0
       ,mapping_name              varchar (255)             not null default 'unknown'
       ,mapping_dependencies      varchar (4000)            not null default 'unknown'
       ,view_name                 varchar (255)             not null default 'unknown'
       ,target_table_name         varchar (255)             not null default 'unknown'
       ,description               varchar (4000)            not null default 'unknown'
       ,development_order         integer                   not null default 0
       ,development_priority      varchar (255)             not null default 'unknown'
       ,development_status        varchar (255)             not null default 'unknown'
       ,developer_assigned        varchar (255)             not null default 'unknown'
       ,estimated_hours           integer                   not null default 0
       ,audit_controls_reqd       varchar (255)             not null default 'unknown'
       ,audit_fields_reqd         varchar (4000)            not null default 'unknown'
       ,refresh_methodology       varchar (4000)            not null default 'unknown'
       ,rec_sel_criteria          varchar (4000)            not null default 'unknown'
       ,condition_in_sq           varchar (4000)            not null default 'unknown'
       ,src_table_names           varchar (4000)            not null default 'unknown'
       ,primary_key               varchar (4000)            not null default 'unknown'
       ,parameter01               varchar (4000)            not null default 'unknown'
       ,parameter02               varchar (4000)            not null default 'unknown'
       ,parameter03               varchar (4000)            not null default 'unknown'
       ,parameter04               varchar (4000)            not null default 'unknown'
       ,parameter05               varchar (4000)            not null default 'unknown'
       ,variable01                varchar (4000)            not null default 'unknown'
       ,variable02                varchar (4000)            not null default 'unknown'
       ,variable03                varchar (4000)            not null default 'unknown'
       ,variable04                varchar (4000)            not null default 'unknown'
       ,variable05                varchar (4000)            not null default 'unknown'
       ,source_connection         varchar (255)             not null default 'unknown'
       ,target_connection         varchar (255)             not null default 'unknown'
       ,err_log_db_connection     varchar (255)             not null default 'unknown'
       ,default_fields            varchar (4000)            not null default 'unknown'
       ,session_names             varchar (4000)            not null default 'unknown'
       ,workflow_names            varchar (4000)            not null default 'unknown'
)
```

4.2.26.2. ctl_mapping_doc_infa_02

This table contains the second portion of data for the definition of Informatica Mappings.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_int_key	The PK Int Key is a sequence number used as the primary key to the table.
mapping_name	This is the name of the Informatica mapping to be produced.
mapping_logic_comments	Where there is some complex logic in the mapping that needs to be further defined and documented this field is available to document such mapping logic comments.
lookup_notes	Where there is some complexity in the lookups to be performed in the mapping the 'Lookup Notes' may be used to document these complexities for the programmer.
data_flows	Where there is some complexity in the Data Flows to be performed in the mapping the 'Data Flows' may be used to document these complexities for the programmer.
notes1	There is room for up to 10 notes to be placed into the Program Specification for the Informatica Programmer.
notes2	There is room for up to 10 notes to be placed into the Program Specification for the Informatica Programmer.
notes3	There is room for up to 10 notes to be placed into the Program Specification for the Informatica Programmer.
notes4	There is room for up to 10 notes to be placed into the Program Specification for the Informatica Programmer.
notes5	There is room for up to 10 notes to be placed into the Program Specification for the Informatica Programmer.
notes6	There is room for up to 10 notes to be placed into the Program Specification for the Informatica Programmer.
notes7	There is room for up to 10 notes to be placed into the Program Specification for the Informatica Programmer.
notes8	There is room for up to 10 notes to be placed into the Program Specification for the Informatica Programmer.
notes9	There is room for up to 10 notes to be placed into the Program Specification for the Informatica Programmer.
notes10	There is room for up to 10 notes to be placed into the Program Specification for the Informatica Programmer.

Example DDL for the table is as follows:

```
create table dbo.ctl_mapping_doc_infa_02 (
        pk_spreadsheet_name       varchar (255)             default 'unknown'
       ,pk_spreadsheet_version    varchar (255)             default 'unknown'
       ,pk_int_key                integer                   not null default 0
       ,mapping_name              varchar (255)             not null default 'unknown'
       ,mapping_logic_comments    varchar (4000)            not null default 'unknown'
       ,lookup_notes              varchar (4000)            not null default 'unknown'
       ,data_flows                varchar (4000)            not null default 'unknown'
       ,notes1                    varchar (4000)            not null default 'unknown'
       ,notes2                    varchar (4000)            not null default 'unknown'
       ,notes3                    varchar (4000)            not null default 'unknown'
       ,notes4                    varchar (4000)            not null default 'unknown'
       ,notes5                    varchar (4000)            not null default 'unknown'
       ,notes6                    varchar (4000)            not null default 'unknown'
       ,notes7                    varchar (4000)            not null default 'unknown'
       ,notes8                    varchar (4000)            not null default 'unknown'
       ,notes9                    varchar (4000)            not null default 'unknown'
       ,notes10                   varchar (4000)            not null default 'unknown'
)
```

4.2.27.ctl_segmentation_control

The SeETL Segmentation Control table is used only by the segmentation engine and the reports from the segmentation engine. It is used to control the Segmentation Engine processing and to be used as an index into the reports for segmentation runs.

Column Name	Function
pk_segment_number	This is an integer key that is used to be able to retrieve the row for queries and to make sure each row has a unique key.
segmentation_type_code	This is the code for the type of segmentation run that this row refers to.
segmentation_run_date	This is the run date of the segmentation run. It is not necessarily the system date.
number_of_days	This is the number of days of transactions that should be read from the detailed data warehouse as input to the segmentation run. It allows a parameterised way of selecting a period of records for consideration of segmentation runs. In conjunction with the segmentation run date it allows segmentation runs to be performed for historical data. Therefore, should the segmentation models change or should the segmentation engine be implemented well after the implementation of the data warehouse it is possible to go back and run the segmentation engine on the historical data.
batch_complete_flag	This is an indicator to tell if the batch is complete or not. 0 for incomplete and 1 for complete.

Example DDL for the table is as follows:

```
create table dbo.ctl_segmentation_control
    (  pk_segment_number            integer     not null    primary key
    ,  segmentation_type_code       varchar (255)
    ,  segmentation_run_date        datetime    not null    default current_timestamp
    ,  number_of_days               integer
    ,  batch_complete_flag          integer
    )
```

4.2.28.ctl_dw_views_doc

In this 3.1.00 release we have made a MASSIVE upgrade to the BI4ALL Data Models. This is the fact that we have now moved all the data models to an Excel Workbook and we can maintain data models in that workbook.

The Data Models can now be maintained in 10 languages over the top of a single physical data model. We provide the only tool (that we know of) to do this. Therefore, for multinational companies they can now roll out the data models to their data warehouses in up to 10 languages and have this all easily and simply maintained via an excel workbook.

The SeETL Design Time User Guide contains more details.

4.2.28.1. ctl_dw_views_doc_01

The first row of the spreadsheet defines the number of languages the data models are available in.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**nl** – Number of Languages.
number_languages	This field defines the number of languages that the base reports will be translated to. It must be a numeric value between 1 and 10 inclusive.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_views_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,number_languages       integer           not null default 0
)
```

4.2.28.2. ctl_dw_views_doc_02

This is a set of rows that defines the target directories to which the view statements will be written.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**dir** – Directory Name for source and target directories.
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
target_directory	This is the target directory where the translated reports will be written for language N.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_views_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,target_directory       varchar (255)     not null default 'unknown'
)
```

4.2.28.3. ctl_dw_views_doc_03

The views that are documented and defined in the data models have short and long descriptions in up to 10 languages defined for them. This set of rows defines the short descriptions for the views.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**vdsd** – View Description Short Description
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
view_owner	The owner of the view. This is documentation to be able to link this view across to the view in the detailed view definition.
view_name	The English name of the view.
submodel	The name of the submodel that the view belongs to.
view_number	This is a unique number given to each view that can be used to describe the view in shorthand. It is used on where clauses to expose the view in many cases as well.
view_short_description	This is the description of the view in the N language.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_views_doc_03 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,view_owner             varchar (255)     not null default 'unknown'
      ,view_name              varchar (255)     not null default 'unknown'
      ,submodel               varchar (255)     not null default 'unknown'
      ,view_number            integer           not null default 0
      ,view_short_description varchar (4000)    not null default 'unknown'
)
```

4.2.28.4. ctl_dw_views_doc_04

The views that are documented and defined in the data models have short and long descriptions in up to 10 languages defined for them. This set of rows defines the long descriptions for the views.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**vdld** – View Description Short Description
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
view_owner	The owner of the view. This is documentation to be able to link this view across to the view in the detailed view definition.
view_name	The English name of the view.
view_long_description	This is the long description of the view in the 0N language.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_views_doc_04 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,view_owner             varchar (255)     not null default 'unknown'
      ,view_name              varchar (255)     not null default 'unknown'
      ,view_long_description  varchar (4000)    not null default 'unknown'
)
```

4.2.28.5. ctl_dw_views_doc_05

The bulk of the information held for views is the view column definitions. These definitions are obviously used to generate the sql statements that are the create view statements over the top of the physical tables which are defined and created using whatever tool the client chooses to use.

These rows allow the view to be created into up to 10 languages and allow the help text for the fields to be defined into up to 10 languages as well. Every effort has been made to make the generation of the views as flexible and as easy as possible while also maintaining the maximum amount of documentation and detail.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**vcd** – View Column Definitions.
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
pk_row_number	This is an integer that is created by the SeETL Design Time application to ensure rows that are loaded have a unique primary key.
row_type	This defines the type of the row that the SeETL [DT] software will interpret the row as. There are some 'helper' values in here to make the code a little easier to write. The valid values are: • **b** – Begin. Each view should have a 'b' on the first line to warn the generator that it is the beginning of a new view. • **m** – Move. This is the 'normal' or default value of a row type. • **d** – derived. If the user wishes to create a 'derived' value as part of a view he is required to place 'd-derived' into the column and then place the derivation in the cell. • **f** – From. The from clause is required to be marked with an 'f' in this column. • **w** – Where. The where clause is required to be marked with a 'w' in this column. • **e** – End. Each view is required to have an 'end' line to tell the generator to generate the end portions of the create view statement. This just made life a little easier.
table_owner	This is the owner of the table and this will be used in the create view statement.
table_name	This is the name of the underlying table that the data is in. It can be the 'meaningless' names that are used by SeETL or the client may change these names to be tables he implements. The underlying tables are only available in one language and they views over the top are then available in 10 languages.
table_column_name	This is the name of the column in the underlying table.
view_owner	This is the owner of the view and this will be used in the create view statement.
view_name	This is the name of the view in the Nth language.
view_column_name	This is the name of the column of the view in the Nth language.
view_column_name_sdesc	This is the short help text for the column in the Nth Language.
view_column_name_ldesc	This is the long help text for the column in the Nth Language.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_views_doc_05 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,pk_row_number          integer           not null default 0
      ,row_type               varchar (255)     not null default 'unknown'
      ,table_owner            varchar (255)     not null default 'unknown'
      ,table_name             varchar (4000)    not null default 'unknown'
      ,table_column_name      varchar (4000)    not null default 'unknown'
      ,view_owner             varchar (255)     not null default 'unknown'
      ,view_name              varchar (255)     not null default 'unknown'
      ,view_column_name       varchar (255)     not null default 'unknown'
      ,view_column_name_sdesc varchar (4000)    not null default 'unknown'
      ,view_column_name_ldesc varchar (4000)    not null default 'unknown'
)
```

4.2.29.ctl_data_model_joins

The SeETL Workbook has been upgraded to support the definition of all joins in the ***BI4ALL Data Models*** into the workbook. This is used to be able to update business objects universes with all the joins needed to be placed into the universe. It also allows SeETL to be able to generate Referential Integrity Constraints into the ***BI4ALL Documentation Database*** that can be re-engineered into various data modeling tools.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
Database_Name	This is the name of the database that contains the data model that these joins are being defined for.
Left_Table_Owner	This is the name of the owner of the table/view that is on the left hand side of the join.
Left_Table	This is the name of the table/view that is on the left hand side of the join.
Left_Table_Column_Name	This is the name of the column that is on the left hand side of the join.
Right_Table_Owner	This is the name of the owner of the table/view that is on the right hand side of the join.
Right_Table	This is the name of the table/view that is on the right hand side of the join.
Right_Table_Column_Name	This is the name of the column that is on the right hand side of the join.
Join_Type	This defines the join type that is to be applied to the join.
Cardinality	This defines the cardinality of the join. Many to one, one to one, one to many.

Example DDL for the table is as follows:

```
create table dbo.ctl_data_model_joins (
       pk_spreadsheet_name    varchar (255)                default 'unknown'
      ,pk_spreadsheet_version varchar (255)                default 'unknown'
      ,pk_row_number          integer                      default 0
      ,Database_Name          varchar     (255)            default 'unknown'
      ,Left_Table_Owner       varchar     (255)            default 'unknown'
      ,Left_Table             varchar     (255)            default 'unknown'
      ,Left_Table_Column_Name varchar     (255)            default 'unknown'
      ,Right_Table_Owner      varchar     (255)            default 'unknown'
      ,Right_Table            varchar     (255)            default 'unknown'
      ,Right_Table_Column_Namevarchar     (255)            default 'unknown'
      ,Join_Type              varchar     (255)            default 'unknown'
      ,Cardinality            varchar     (255)            default 'unknown'
)
```

4.2.30.ctl_dw_sps_doc

There are two separate functions for Stored Procedures. They are the generation and manipulation of Stored Procedures and then the documentation of Stored Procedures. We have developed a solution for managing Stored Procedures in many languages and then deploying them to the database of choice for the client. Very cool.

BI4ALL Analysis is a new product that we are working on. Currently there are 70+ Microsoft Report Services Reports in the product. These reports get data from the Data Warehouse via the ***'Data Access Layer'***. The ***Data Access Layer*** is a library of about 60,000 lines of Stored Procedures. We initially released the Stored Procedures for SQL Server only. We did not want to undertake the massive development effort of developing and deploying multiple sets of stored procedures to multiple databases using multiple languages until such time as it was necessary.

In order to deliver stored procedures in many database language dialects we elected to upgrade SeETL to be able to read a set of stored procedures defined in a workbook and to apply filtering and directives to those pieces of text to produce separate files of output. Each separate file of output would be the stored procedure in the specific language for a specific database. The input file would contain all the different text required for all the different languages.

It is felt that such a solution, a single input text file, a generator/translator, and separate output files is a perfectly adequate way to manage different sets of stored procedures so as to allow IBI to maintain a single code base but to also facilitate the easy deployment of the ***Data Access Layer*** to any of the supported databases.

The Generate Stored Procedures tab that supports the translation of ***BI4ALL*** Analysis Stored Procedures into the different database stored procedure languages is relatively complex because of the amount of information that is required to be captured.

4.2.30.1. ctl_dw_sps_doc_01

The first row of the spreadsheet defines the number of languages the Stored Procedure is stored in.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
number_languages	This field defines the number of languages that the input stored procedure file will emit as the generation process takes place. It must be a numeric value between 1 and 10 inclusive.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sps_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,number_languages       integer           not null default 0
)
```

4.2.30.2. ctl_dw_sps_doc_02

This table records the target directories that the translated Stored Procedures will be written to.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
target_directory	This is the target directory where the generated and translated stored procedures will be written for language N.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sps_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,target_directory       varchar (255)     not null default 'unknown'
)
```

4.2.30.3. ctl_dw_sps_doc_03

There is no requirement for the stored procedure to be stored in the same file name for the different language versions. So SeETL includes the ability to name the input file to be read and then associate that input file with different output file names in different directories. This seemed like a useful feature to add and it replicated the similar feature in the report translation process.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
stored_procedure_name	The name of the Stored Procedure to be translated/generated.
source_file_name	The name of the file the Stored Procedure to be translated/generated is stored in.
target_file_name	The name of the file the Stored Procedure to be translated/generated is to be written to.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sps_doc_03 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,stored_procedure_name  varchar (255)     not null default 'unknown'
      ,source_file_name       varchar (255)     not null default 'unknown'
      ,target_file_name       varchar (255)     not null default 'unknown'
)
```

4.2.30.4. ctl_dw_sps_doc_04

The languages set parameters tell the generator / translator which languages should be considered turned 'on' or 'off' as well as provide some language specific parameters to apply to the generation / translation process. All databases have slightly different aspects to them and these aspects needs to be provided for using the 'Language Set' directives.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**lds** – Language Directives Set.
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
language_name	This is an integer listing the language number. It is only required to provide a simply key to the documentation table that this set of directives is written to.
language_status	This is a flag set to 'On' or 'Off' for a language. Please take care that this matches with the number of languages directive and that the number of languages the user wishes to generate/translate occur in the top 'n' rows of this set of directive entries. There is no allowance for turning directives on or off up and down the list of languages.
language_table_owner	Inside the text the string **##languagetableowner##** will be used to denote the language table owner parameter. The different databases use different default table owners and most companies also have standards about the name of the table owner for tables. Some companies insist that the table owner is explicitly named and we have adopted the idea that the table owner must be explicitly named. It is just safer that way. The generator/translator uses this input directive and it replaces all instances of **##languagetableowner##** in the text with this parameter.
language_parameter_prefix	Inside the text the string **##languageparameterprefix##** will be used to denote the prefix to be used for parameters named in the stored procedure. In SQL Serve the @ symbol is used to denote a prefix. In Netezza no symbol is used but the alias for the variable name can not be the same name as the name of an existing field. So a prefix is required to make sure no clashes of field names and variable names occurs The generator/translator uses this input directive and it replaces all instances of **## languageparameterprefix ##** in the text with this parameter.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sps_doc_04 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,language_name          varchar (255)     not null default 'unknown'
      ,language_status        varchar (255)     not null default 'unknown'
      ,language_table_owner   varchar (255)     not null default 'unknown'
      ,language_parameter_prefix    varchar (255)     not null default 'unknown'
)
```

4.2.30.5. ctl_dw_sps_doc_05

The generation/translation process allows for more directives that are applied across all languages. These directives work very much like C 'defines' statements. A directive is either 'On' or 'Off'. If the directive is 'on' those lines of text marked by the directives will be emitted to ALL languages. Similarly, if the directive 'All:on' is set to on then the text will be emitted to ALL output language files.

It is expected that as time goes on we may find need for more directives than are currently seen as being required.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**ds** – Directives Set.
pk_directive_number	This is an integer listing the directive number. It is only required to provide a simply key to the documentation table that this set of directives is written to.
directive_name	Each of the directives have names. The list of current directive names and their meanings is given below: • RedundantJoins All template stored procedures have all the dimension tables they join to already pre-joined in the text of the stored procedure. This is to save the client time and effort when actually implementing the stored procedures. However, in many cases these joins are not needed in the stored procedures for reports or templates. They are there to make adding joins to new dimensions easier. As such they can be described as 'redundant'. However, even though the joins may be redundant they will cause processing to occur if they are submitted to the database. Hence, there is a need to be able to easily turn them on and off. This is the purpose of this directive. To easily turn redundant joins and the listing of the table name in the where clause on and off more easily. • SQLServerpartitioning In SQL Server 2005 Enterprise Edition there was still only ONE partitioning column. Therefore, IBI was required to come up with a better mechanism for partitioning for our clients on SQL Server. We did this by creating a two part key based on company reporting structure and time. The partitions were broken up into 'country by time' typically. This provided enough flexibility and made the partitions small enough to get decent performance out of the databases. However, the stored procedures needed to be told about this feature and it needed to be embedded in the SQL for the stored procedure. Obviously, on other databases, this feature needs to be turned off. This is the purpose of this directive. To turn on and off the feature that enables multi-part petitioning for SQL Server.
directive_status	This is set to the literals "On" or "Off

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sps_doc_05 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_directive_number    integer           not null default 0
      ,directive_name         varchar (255)     not null default 'unknown'
      ,directive_status       varchar (255)     not null default 'unknown'
)
```

4.2.30.6. ctl_dw_sps_doc_06

BI4ALL Analysis is a new product that we are working on. Currently there are 70+ Microsoft Report Services Reports in the product. These reports get data from the Data Warehouse via the ***'Data Access Layer'***. The ***Data Access Layer*** is a library of about 60,000 lines of Stored Procedures. We initially released the Stored Procedures for SQL Server only. We did not want to undertake the massive development effort of developing and deploying multiple sets of stored procedures to multiple databases using multiple languages until such time as it was necessary.

One issue with having a large library of stored procedures is documenting them in such a way that the dictionary for the stored procedures is easily available and searchable for all those who would wish to understand what is in the stored procedure.

To the current point in time only IBI Development Staff and developers at business partners have been required to learn these stored procedures. And they have had to do it the 'normal' (read hard) way. They have to sit down and read the source code and learn it. In most cases the people involved are then involved with the stored procedures for some years so the learning curve has not been significant compared to the time using the stored procedures.

This is about to change. We are about to make the stored procedures 'open' and 'available' to business users who wish to use them to do such things as call them from inside a query window and cut/paste the results into excel spreadsheets. We are expecting to be able to run stored procedures from inside excel spreadsheets in the not too distant future as Microsoft makes excel more open to it's own database as well as others.

Further, we expect that now that the stored procedures will be ported to Netezza that there will be a need to integrate the stored procedures with business objects. This will mean business objects report writers will need documentation for these stored procedures.

In order to deliver documentation for stored procedures we have developed a simple catalogue that will be extended over time. We are going to go about the process of filling up this catalogue for all the existing procedures as well as providing it to clients to document their own stored procedures. Tools to document stored procedures have been poor to date. This tool will make some significant contribution to our clients to document their stored procedures.

The Document Stored Procedures tab that supports the documentation of ***BI4ALL Analysis*** Stored Procedures is relatively complex because of the amount of information that is required to be captured.

The ***Document Stored Procedures Reports*** tab contains the following fields:

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**nl** – Number of Languages.
number_languages	This field defines the number of languages that the stored procedure documentation will exist in and be loaded in to the catalogue database. Usually it will only be English but the ability has been created to deploy in multiple languages for descriptions.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sps_doc_06 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,number_languages       integer           not null default 0
)
```

4.2.30.7. ctl_dw_sps_doc_07

The stored procedures that are documented and defined in this workbook have short and long descriptions in up to 10 languages defined for them. This set of rows defines the short descriptions for the Stored Procedures.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**spsd** – Stored Procedure Description Short Description
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
procedure_owner	The owner of the Stored Procedure. This is documentation to be able to link this Stored Procedure across to the Stored Procedure Name in the detailed Stored Procedure definition.
procedure_name	The name of the Stored Procedure in Nth language.
submodel	The name of the submodel that the Stored Procedure belongs to.
procedure_number	This is a unique number given to each Stored Procedure that can be used to describe the Stored Procedure in shorthand. It is used simply to make it easier to refer to Stored Procedures and to use as a key in the documentation database.
procedure_short_description	This is the description of the Stored Procedure in the Nth language.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sps_doc_07 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,pk_row_number          integer           not null default 0
      ,procedure_owner        varchar (255)     not null default 'unknown'
      ,procedure_name         varchar (255)     not null default 'unknown'
      ,submodel               varchar (255)     not null default 'unknown'
      ,procedure_number       integer           not null default 0
      ,procedure_short_description  varchar (4000)    not null default 'unknown'
)
```

4.2.30.8. ctl_dw_sps_doc_08

The Stored Procedures that are documented and defined in this workbook have short and long descriptions in up to 10 languages defined for them. This set of rows defines the long descriptions for the Stored Procedures.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**spld** – Stored Procedure Description Long Description
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
procedure_owner	The owner of the Stored Procedure. This is documentation to be able to link this Stored Procedure across to the Stored Procedure Name in the detailed Stored Procedure definition.
procedure_name	The English name of the Stored Procedure.
procedure_long_description	This is the long description of the Stored Procedure in the Nth language.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sps_doc_08 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,pk_row_number          integer           not null default 0
      ,procedure_owner        varchar (255)     not null default 'unknown'
      ,procedure_name         varchar (255)     not null default 'unknown'
      ,procedure_long_description   varchar (4000)    not null default 'unknown'
)
```

4.2.30.9. ctl_dw_sps_doc_09

To be able to properly use a Stored Procedure the user calling the stored procedure must know the input fields that the stored procedure is expecting to receive. This section documents the input fields of the stored procedure in enough depth that the end user has a reasonable chance to be able to properly call the stored procedure.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**spid** – Stored Procedure Input Definitions.
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
row_type	This defines the type of the row that the SeETL[DI] software will interpret the row as. There are some 'helper' values in here to make the code a little easier to write. The valid values are: • **b** – Begin. Each view should have a 'b' on the first line to warn the generator that it is the beginning of a new view. • **m** – Move. This is the 'normal' or default value of a row type. • **e** – End. Each view is required to have an 'end' line to tell the generator to generate the end portions of the create view statement. This just made life a little easier.
procedure_owner	The owner of the Stored Procedure. This is documentation to be able to link this Stored Procedure across to the Stored Procedure Name in the detailed Stored Procedure definition.
procedure_name	The name of the Stored Procedure in the Nth language.
input_column_name	This is the name of the input columns to the stored procedure.
input_column_data_type	This is a text description of the input column data type.
input_column_sdesc	This is the short help text of the input column in language N.
input_column_ldesc	This is the long help text of the input column in language N.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sps_doc_09 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,pk_row_number          integer           not null default 0
      ,row_type               varchar (255)     not null default 'unknown'
      ,procedure_owner        varchar (255)     not null default 'unknown'
      ,procedure_name         varchar (255)     not null default 'unknown'
      ,input_column_name      varchar (255)     not null default 'unknown'
      ,input_column_data_type varchar (255)     not null default 'unknown'
      ,input_column_sdesc     varchar (4000)    not null default 'unknown'
      ,input_column_ldesc     varchar (4000)    not null default 'unknown'
)
```

4.2.30.10.ctl_dw_sps_doc_10

To be able to properly use a Stored Procedure the user calling the stored procedure must know the output fields that the stored procedure is expecting to receive. This section documents the output fields of the stored procedure in enough depth that the end user has a reasonable chance to be able to properly take the data from the stored procedure and place it onto a report or into some other front end reporting tool.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**spod** – Stored Procedure Output Definitions.
pk_language_number	This field defines the language number (1 to 10) of the record. 1 is defaulted to mean English. The other 9 languages must be defined by the client.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
row_type	This defines the type of the row that the SeETL[DT] software will interpret the row as. There are some 'helper' values in here to make the code a little easier to write. The valid values are: • **b** – Begin. Each view should have a 'b' on the first line to warn the generator that it is the beginning of a new view. • **m** – Move. This is the 'normal' or default value of a row type. • **e** – End. Each view is required to have an 'end' line to tell the generator to generate the end portions of the create view statement. This just made life a little easier.
procedure_owner	The owner of the Stored Procedure. This is documentation to be able to link this Stored Procedure across to the Stored Procedure Name in the detailed Stored Procedure definition.
procedure_name	The name of the Stored Procedure in the Nth language.
output_column_name	This is the name of the Output columns to the stored procedure.
output_column_data_type	This is a text description of the Output column data type.
output_column_sdesc	This is the short help text of the Output column in language N.
output_column_ldesc	This is the long help text of the Output column in language N.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sps_doc_10 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_language_number     integer           not null default 0
      ,pk_row_number          integer           not null default 0
      ,row_type               varchar (255)     not null default 'unknown'
      ,procedure_owner        varchar (255)     not null default 'unknown'
      ,procedure_name         varchar (255)     not null default 'unknown'
      ,output_column_name     varchar (255)     not null default 'unknown'
      ,output_column_data_type varchar (255)    not null default 'unknown'
      ,output_column_sdesc    varchar (4000)    not null default 'unknown'
      ,output_column_ldesc    varchar (4000)    not null default 'unknown'
)
```

4.2.31.ctl_dw_other_views_doc

The Other Views tab is used to store definitions of views that the Architect might like to define and which are not generated by SeETL in some other way.

In every implementation it has happened that there are some views which get created to make some portion of work that little bit easier. These views have normally just been created in text documents. However, in text documents they are not particularly 'visible' and this broke with the concept of 'You and see your ETL' which is a design point, and the basis for the name, of SeETL.

It was decided to make it possible to put all 'Other Views' into the SeETL^{DT} Workbook and publish them via the portal just to 'close the gap' on having all portions available. The Architect can decide for him/her self whether he/she will use these other views for this style of documentation.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_row_type	The type of the field being defined. It may be: • 'M' – Move • 'F – From • 'W' – Where • 'G' – Group By • 'O' – Order by
table_owner	This is the owner of the table over which the view is being placed.
table_name	This is the name of the table over which the view is being placed.
table_column_name	This is the name of the column in the table over which the view is being placed.
view_owner	This is the name of the owner of the view that is being created. It is not necessarily the same as the name of the owner of the table.
view_name	This is the name of the view being created.
view_column_name	This is the name of the column being created in the view.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_other_views_doc (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,table_owner            varchar (255)     not null default 'unknown'
      ,table_name             varchar (255)     not null default 'unknown'
      ,table_column_name      varchar (255)     not null default 'unknown'
      ,view_owner             varchar (255)     not null default 'unknown'
      ,view_name              varchar (255)     not null default 'unknown'
      ,view_column_name       varchar (255)     not null default 'unknown'
)
```

4.2.32.ctl_dw_compound_views_doc

The Compound Views tab is used to store definitions of views that require some form of compound statement such as the 'union all' statement that the Architect might like to define. Such statements are not supported by other tabs such as the mappings or 'Other Views' tabs.

In occasional implementations it has happened that there are some 'union views' which get created to make some portion of work that little bit easier. These views have normally just been created in text documents. However, in text documents they are not particularly 'visible' and this broke with the concept of 'You can see your ETL' which is a design point of SeETL.

It was decided to make it possible to put all 'Compound Views' into the SeETL^{DT} Workbook and publish them via the portal just to 'close the gap' on having all portions available. The Architect can decide for him/her self whether he/she will use these other views for this style of documentation.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_row_type	The type of the field being defined. It may be: • 'B' – Begin. Each new view must begin with a 'B' for begin line. • 'E' – End. Each view must have a null 'E' for end line. • 'M' – Move • 'D' – Derived. • 'C' – Compound. On a line denoting the 'compound' view the type is 'C'. • 'F – From • 'W' – Where • 'G' – Group By • 'O' – Order by
table_owner	This is the owner of the table over which the view is being placed.
table_name	This is the name of the table over which the view is being placed. When the 'Type' is 'C' for 'Compound View' this field contains the compound view statement.
table_column_name	This is the name of the column in the table over which the view is being placed.
view_owner	This is the name of the owner of the view that is being created. It is not necessarily the same as the name of the owner of the table.
view_name	This is the name of the view being created.
view_column_name	This is the name of the column being created in the view.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_compound_views_doc (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,table_owner            varchar (255)     not null default 'unknown'
      ,table_name             varchar (255)     not null default 'unknown'
      ,table_column_name      varchar (255)     not null default 'unknown'
      ,view_owner             varchar (255)     not null default 'unknown'
      ,view_name              varchar (255)     not null default 'unknown'
      ,view_column_name       varchar (255)     not null default 'unknown'
)
```

4.2.33.ctl_dw_sql_stmts

SeETL[DT] has now been updated to be able to load SQL Statements into a target table to be processed at a later date by the scheduler. This is a ***MAJOR*** improvement in security and auditability.

The idea behind this feature is this. We upgraded the workbook to be able to generate SQL as the ETL subsystem. This left us with HUNDREDS of SQL fragments that need to be managed. Some clients have repositories to maintain these snippets of SQL and some do not.

Those that do not have a repository to manage all these snippets of code are then faced with the normal security and auditability problems of text source code. We placed many comments such as userid, machine name, timestamps etc at the front of each snippet of code. This helps to identify the person who generated the SQL. However it does not version the SQL and give the client a chance to 'back out to a prior version' or things like this.

To provide versioning, increased security and auditability we are now placing the SQL generated into a table. SeETL[DT] will always turn off the 'current flag' for the existing statement and then insert the new SQL statement into the database.

Since SQL Server can only support fields of 8000 bytes the SQL Statement is broken into up to 4 pieces of 7,900 bytes each. This allows for long statements.

Each time the statement is loaded from a file the old version will be closed off and the new version will be written into the database with a timestamp as well as details of who made the change to the statement. Later, when the statement is executed the first 8,000 bytes of the statement is logged along with the timestamp. In this way it is possible to know exactly what statement was executed at any given time save for malicious efforts to do otherwise by the DBA of the production database.

Column Name	Function
pk_spreadsheet_name	This is the name of the spreadsheet that is used to cause the sql statement to be loaded into this table.
pk_spreadsheet_version	This is the version of the spreadsheet that is used to cause the sql statement to be loaded into this table.
pk_row_number	Each load action is given a sequential row number starting from zero.
pk_load_timestamp	This is the timestamp of the row being loaded into the database. The timestamp makes the key unique.
current_flag	Rows may never be deleted from this table by the SeETL[DT] application. Only an administrator to the table may perform deletes if that is how the security of the table is set.
directory_name	This is the name of the directory that the sql in the file is loaded from. The directory is stored separately so that it is possible to easily build reports based on directory.
sql_file_name	This is the name of the file that the sql is loaded from.
full_sql_file_name	The full directory and file name is also stored to make it simple for the program to perform a direct match on this field with the SQLFileName parameter.
sql_file_win_user	This is the name of the authenticated windows user that was running the SeETL[DT] application to load this sql from the file into the database.
sql_file_win_workstation	This is the name of the workstation that was running the SeETL[DT] application to load this sql from the file into the database.
sql_file_win_domain	This is the name of the authenticated windows domain that was running the SeETL[DT] application to load this sql from the file into the database.
sql_file_os_version	This is the name of the authenticated windows operating system that was running the SeETL[DT] application to load this sql from the file into the database.
sql_file_ctl_current_directory	This is the name of the directory out of which SeETL[DT] was running when it loaded the statement into the database.
sql_file_dot_net_version	This is the name of the authenticated windows dot net version that was running the SeETL[DT] application to load this sql from the file into the database.
sql_file_CTL_version_date	This is the build date of the SeETL[DT] version so that the client can tell exactly what version of SeETL[DT] they are running. We have not typically reversions SeETL[DT] every time we have made updates.
sql_statement_01	This is the first 8000 characters of the SQL Statement.
sql_statement_02	This is the second 8000 characters of the SQL Statement.

sql_statement_03	This is the third 8000 characters of the SQL Statement.
sql_statement_04	This is the fourth 8000 characters of the SQL Statement.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sql_stmts (
       pk_spreadsheet_name       varchar (255)       default 'unknown'
      ,pk_spreadsheet_version    varchar (255)       default 'unknown'
      ,pk_row_number             integer             not null default 0
      ,pk_load_timestamp         datetime            not null   default current_timestamp
      ,current_flag              integer             not null default 0
      ,directory_name            varchar (4000)      not null default 'unknown'
      ,sql_file_name             varchar (4000)      not null default 'unknown'
      ,full_sql_file_name        varchar (4000)      not null default 'unknown'
      ,sql_file_win_user         varchar (255)       default 'unknown'
      ,sql_file_win_workstation  varchar (255)       default 'unknown'
      ,sql_file_win_domain       varchar (255)       default 'unknown'
      ,sql_file_os_version       varchar (255)       default 'unknown'
      ,sql_file_ctl_current_directory         varchar (4000)default 'unknown'
      ,sql_file_dot_net_version  varchar (255)       default 'unknown'
      ,sql_file_CTL_version_date       varchar (255)default 'unknown'
      ,sql_statement_01          varchar (8000)      not null default 'unknown'
      ,sql_statement_02          varchar (8000)      not null default 'unknown'
      ,sql_statement_03          varchar (8000)      not null default 'unknown'
      ,sql_statement_04          varchar (8000)      not null default 'unknown'
)
```

4.2.34.ctl_dw_sql_stmt_log

This is the log file that is written to when SQL Statements are executed by the Execute SQL Statement and Logging Utility.

The following table explains the values of each column and the functions available.

Column Name	Function
pk_spreadsheet_name	This is the name of the spreadsheet that is used to cause the sql statement to be loaded into this table.
pk_spreadsheet_version	This is the version of the spreadsheet that is used to cause the sql statement to be loaded into this table.
pk_row_number	Each load action is given a sequential row number starting from zero.
pk_load_timestamp	This is the timestamp of the row being loaded into the database. The timestamp makes the key unique.
pk_batch_number	This is the number of the batch that the SQL statement was executed in. By storing this it will be possible to observe the differences in elapsed times or rows processed across different batches.
pk_start_stmt_timestamp	The start timestamp of the sql statement. It is made part of the key to make the key unique.
end_stmt_timestamp	The stop timestamp of the statement. Therefore the elapsed time of the statement can be worked out for reporting purposes.
directory_name	This is the name of the directory that the sql in the file is loaded from. The directory is stored separately so that it is possible to easily build reports based on directory.
sql_file_name	This is the name of the file that the sql is loaded from.
full_sql_file_name	The full directory and file name is also stored to make it simple for the program to perform a direct match on this field with the SQLFileName parameter.
number_rows_affected	This is the number of rows affected by the SQL statement as returned by the database manager.
sql_code	This is the SQL Code returned by the database manager.
sql_statement	This is the first 8,000 bytes of the SQL statement purely for logging purposes.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_sql_stmt_log (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_load_timestamp      datetime          not null   default current_timestamp
      ,pk_batch_number        integer           not null
      ,pk_start_stmt_timestamp      datetime    not null   default current_timestamp
      ,end_stmt_timestamp     datetime          not null   default current_timestamp
      ,directory_name         varchar (4000)    not null default 'unknown'
      ,sql_file_name          varchar (4000)    not null default 'unknown'
      ,full_sql_file_name     varchar (4000)    not null default 'unknown'
      ,number_rows_affected   integer           not null default 0
      ,sql_code               integer           not null default 0
      ,sql_statement          varchar (8000)    not null default 'unknown'
)
```

4.2.35. Data Profiling Worksheet

The SeETL[DT] has now been updated to be able to generate Data Profiling SQL that can be submitted against any ANSI SQL compliant database. Data Profiling is an area where the vendors have done a relatively poor job. The engines simply don't do that much that is of value over and above what a bit of experience will bring to the table. We decided to create some simple data profiling algorithms and we will add more and more complex algorithms as we see how people start to use the simple ones.

4.2.35.1. ctl_dw_dp_row_counts

The tables that the results of data profiling are loaded into are as follows. Notice that all the rows have the name of the spreadsheet that generated them, the timestamp of the inserted row and the current flag. This allows all Data Profiling to be versioned.

The first table maintains the number of rows in tables. It is an easy way of maintaining row counts over time. This is useful in the data profiling of data for development of a project. It is also useful information to have for the staging area and data warehouse over time.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_run_timestamp	This is the timestamp of the row that was inserted into this table by the SQL generated by the Data Profiling SQL Generation Facility. This is included in the primary key to make the primary key unique over many versions of the inserted rows.
current_flag	This is a current flag that is set to "0" for a non current row and "1" for the current row.
table_owner	This is the name of the owner of the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
table_name	This is the name of the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
number_rows	This is the number of rows found in the table.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_dp_row_counts (
        pk_spreadsheet_name          varchar (255)               default 'unknown'
       ,pk_spreadsheet_version       varchar (255)               default 'unknown'
       ,pk_row_number                integer                     not null default 0
       ,pk_run_timestamp             datetime                    not null   default current_timestamp
       ,current_flag                 integer                     not null default 0
       ,table_owner                  varchar (255)               not null default 'unknown'
       ,table_name                   varchar (255)               not null default 'unknown'
       ,number_rows                  integer                     not null default 0
)
```

4.2.35.2. ctl_dw_dp_col_counts_01

This table stored the number of unique values found in a column and not the values themselves. It is always useful to count the number of unique values and divide by the number of rows to see how much repetition there basically is in the data.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_run_timestamp	This is the timestamp of the row that was inserted into this table by the SQL generated by the Data Profiling SQL Generation Facility. This is included in the primary key to make the primary key unique over many versions of the inserted rows.
current_flag	This is a current flag that is set to “0” for a non current row and “1” for the current row.
table_owner	This is the name of the owner of the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
table_name	This is the name of the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
column_name	This is the name of the column in the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
column_number	This is the ordinal number of the column in the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
num_unique_values	This is the number of unique values discovered in this column in the table/view.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_dp_col_counts_01 (
         pk_spreadsheet_name          varchar (255)               default 'unknown'
        ,pk_spreadsheet_version       varchar (255)               default 'unknown'
        ,pk_row_number                integer                     not null default 0
        ,pk_run_timestamp             datetime                    not null   default current_timestamp
        ,current_flag                 integer                     not null default 0
        ,table_owner                  varchar (255)               not null default 'unknown'
        ,table_name                   varchar (255)               not null default 'unknown'
        ,column_name                  varchar (255)               not null default 'unknown'
        ,column_number                integer                     not null default 0
        ,num_unique_values            integer                     not null default 0
)
```

4.2.35.3. ctl_dw_dp_col_counts_02

This table stores the ACTUAL unique values in the column. Note that the value itself is stored in a string field of 8,000 bytes. If the values in the column are unique the field 'candidate key flag' is set to 1 to notify the database designer that this might actually be useful as a candidate key.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_run_timestamp	This is the timestamp of the row that was inserted into this table by the SQL generated by the Data Profiling SQL Generation Facility. This is included in the primary key to make the primary key unique over many versions of the inserted rows.
current_flag	This is a current flag that is set to "0" for a non current row and "1" for the current row.
table_owner	This is the name of the owner of the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
table_name	This is the name of the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
column_name	This is the name of the column in the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
column_number	This is the ordinal number of the column in the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
unique_value	This is a string containing the unique values of the field found in the data.
num_occurrences	This is the number of occurrences of this unique value. If it is 1 then it was only found once and it becomes a candidate key.
candidate_key_flag	The candidate key flag is se to 0 if this unique value in this field is found more than once. It will be set to 1 if this value was only ever found once.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_dp_col_counts_02 (
        pk_spreadsheet_name          varchar (255)               default 'unknown'
       ,pk_spreadsheet_version       varchar (255)               default 'unknown'
       ,pk_row_number                integer                     not null default 0
       ,pk_run_timestamp             datetime                    not null   default current_timestamp
       ,current_flag                 integer                     not null default 0
       ,table_owner                  varchar (255)               not null default 'unknown'
       ,table_name                   varchar (255)               not null default 'unknown'
       ,column_name                  varchar (255)               not null default 'unknown'
       ,column_number                integer                     not null default 0
       ,unique_value                 varchar (8000)              not null default 'unknown'
       ,num_occurrences              integer                     not null default 0
       ,candidate_key_flag           integer                     not null default 0
)
```

4.2.35.4. ctl_dw_dp_col_counts_03

This table stores statistics like min, max, average, mode etc for columns. Obviously this only makes sense with numeric columns.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_run_timestamp	This is the timestamp of the row that was inserted into this table by the SQL generated by the Data Profiling SQL Generation Facility. This is included in the primary key to make the primary key unique over many versions of the inserted rows.
current_flag	This is a current flag that is set to “0” for a non current row and “1” for the current row.
table_owner	This is the name of the owner of the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
table_name	This is the name of the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
column_name	This is the name of the column in the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
column_number	This is the ordinal number of the column in the table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
col_average	The result of applying the average function of SQL to this field.
col_mode	The result of applying the mode function of SQL to this field.
col_min	The result of applying the min function of SQL to this field.
col_max	The result of applying the min function of SQL to this field.
col_std_deviation	The result of applying the standard deviaton function of SQL to this field.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_dp_col_counts_03 (
        pk_spreadsheet_name          varchar (255)               default 'unknown'
       ,pk_spreadsheet_version       varchar (255)               default 'unknown'
       ,pk_row_number                integer                     not null default 0
       ,pk_run_timestamp             datetime                    not null   default current_timestamp
       ,current_flag                 integer                     not null default 0
       ,table_owner                  varchar (255)               not null default 'unknown'
       ,table_name                   varchar (255)               not null default 'unknown'
       ,column_name                  varchar (255)               not null default 'unknown'
       ,column_number                integer                     not null default 0
       ,col_average                  decimal (38,12)             not null default 0
       ,col_mode                     decimal (38,12)             not null default 0
       ,col_min                      decimal (38,12)             not null default 0
       ,col_max                      decimal (38,12)             not null default 0
       ,col_std_deviation            decimal (38,12)             not null default 0
)
```

4.2.35.5. ctl_dw_dp_col_counts_04

The final table is trying to match different columns in different tables to guess at joins at the data exploration time. The query to populate this table is, of course, dependent on actually storing all the unique values in the dictionary. Such information on possible joins is some of the most useful information that can be gleaned from the data itself.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_run_timestamp	This is the timestamp of the row that was inserted into this table by the SQL generated by the Data Profiling SQL Generation Facility. This is included in the primary key to make the primary key unique over many versions of the inserted rows.
current_flag	This is a current flag that is set to “0” for a non current row and “1” for the current row.
left_table_owner	This is the name of the owner of the LEFT table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
left_table_name	This is the name of the LEFT table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
left_column_name	This is the name of the column in the LEFT table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
left_column_number	This is the ordinal number of the column in the LEFT table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
left_unique_value	This is a string containing the unique values of the field found in the data in the LEFT table.
left_num_occurrences	This is the number of occurrences of this unique value in the LEFT table. If it is 1 then it was only found once and it becomes a candidate key.
left_candidate_key_flag	The candidate key flag is se to 0 if this unique value in this field is found more than once. It will be set to 1 if this value was only ever found once. This is for the LEFT table.
right_table_owner	This is the name of the owner of the RIGHT table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
right_table_name	This is the name of the RIGHT table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
right_column_name	This is the name of the column in the RIGHT table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
right_column_number	This is the ordinal number of the column in the RIGHT table/view that is to have Data Profiling SQL generated by the Data Profiling Generation Facility.
right_num_occurrences	This is the number of occurrences of this unique value in the RIGHT table. If it is 1 then it was only found once and it becomes a candidate key.
right_candidate_key_flag	The candidate key flag is se to 0 if this unique value in this field is found more than once. It will be set to 1 if this value was only ever found once. This is for the RIGHT table.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_dp_col_counts_04 (
        pk_spreadsheet_name          varchar (255)               default 'unknown'
       ,pk_spreadsheet_version       varchar (255)               default 'unknown'
       ,pk_row_number                integer                     not null default 0
       ,pk_run_timestamp             datetime                    not null   default current_timestamp
       ,current_flag                 integer                     not null default 0
       ,left_table_owner             varchar (255)               not null default 'unknown'
       ,left_table_name              varchar (255)               not null default 'unknown'
       ,left_column_name             varchar (255)               not null default 'unknown'
       ,left_column_number           integer                     not null default 0
       ,left_unique_value            varchar (255)               not null default 'unknown'
       ,left_num_occurrences         integer                     not null default 0
       ,left_candidate_key_flag      integer                     not null default 0
       ,right_table_owner            varchar (255)               not null default 'unknown'
       ,right_table_name             varchar (255)               not null default 'unknown'
       ,right_column_name            varchar (255)               not null default 'unknown'
       ,right_column_number          integer                     not null default 0
       ,right_num_occurrences        integer                     not null default 0
       ,right_candidate_key_flag     integer                     not null default 0
)
```

4.2.36.ctl_dw_tables_doc

The SeETL[DT] Workbook has now been updated to be able to store the ***BI4ALL Data Model Tables*** in the workbook as well. This is a major upgrade to SeETL[DT] and further represents our commitment and ability to place as much as is possible into the workbook and to manage as much of the development process as possible out of the workbook with the complementary SQL Server database and reports.

The ***BI4ALL Data Model Tables*** have previously been delivered in only text format and in only the English Language. This was a conscious decision as the time and effort to maintain data models in tools like Erwin and PowerDesigner is very great.

The ***BI4ALL Data Model Tables*** now come as a separate workbook that the client can then copy/change/load and generate tables from as is desired. We will continue to release upgrades to these models and it is up to the client to manage the upgrade process based on the dictionary provided.

The ***BI4ALL Data Model Tables*** tab that supports the translation of definitions of tables into scripts that can be executed against a variety of databases is relatively complex because of the amount of information that is required to be captured.

4.2.36.1. ctl_dw_tables_doc_01

The first row of the spreadsheet defines the Target Directory that the create table statements will be generated into.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
target_directory	This field contains the name of the target directory for the create table scripts. It is recommended that it contain no quotes and no blanks as these have not been tested and will not be supported.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_tables_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,target_directory       varchar (4000)    not null default 'unknown'
)
```

4.2.36.2. ctl_dw_tables_doc_02

The bulk of the information held for tables is the table definitions rows. These definitions are obviously used to generate the sql statements that are the create table statements.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_row_type	This defines the type of the row that the SeETL[DT] software will interpret the row as. There are some 'helper' values in here to make the code a little easier to write. The valid values are: • **b** – Begin. Each table should have a 'b' on the first line to warn the generator that it is the beginning of a new view. • **m** – Move. This is the 'normal' or default value of a row type. • **n** – eNd. This indicates the last row of the definitions of the columns for the table. The rows that follow will contain special clauses on a database specific basis. • **s** – Special. All databases have different clauses that can be used at the end of the create table statement to do things like create relationships for tables spaces or storage spaces and things like that. The way this have been implemented in the SeETL[DT] Workbook is to allow free form text in cells that will be directly replicated into the create table statements. • **e** – End. Each table is required to have an 'end' line to tell the generator to generate the end portions of the create table statement. This just made life a little easier.
table_owner	This is the owner of the table and this will be used in the create table statement.
table_name	This is the name of the table that the data is in. It can be the 'meaningless' names that are used by ***BI4ALL*** or the client may change these names to be tables he implements. The underlying tables are only available in one language and the views over the top are then available in 10 languages.
table_column_name	This is the name of the column in the table.
table_column_number	This is the ordinal number of the column in the table.
column_data_type_deploy	This is the data type that will be accepted by the database that the table is being deployed to. The deployment database may be a different database to the documentation database. For example a client may deploy to Netezza or Oracle but document to SQL Server to gain the benefits of the already provided Report Services Reports.
column_data_type_document	This is the data type that will be accepted by the database that the table is being documented to.
column_null_clause_deploy	This is the null clause that will be accepted by the database that the table is being deployed to. It may be left empty of no null clause is desired. However, in most cases it is recommended that a null clause is specified.
column_null_clause_document	This is the null clause that will be accepted by the database that the table is being documented to. It may be left empty of no null clause is desired. However, in most cases it is recommended that a null clause is specified.
column_other_clauses	This is a field that is made available to provide any other clauses that need to be created for the table to be properly created onto the deployment or documentation database. In some cases it might be necessary to store multiple rows with multiple instances of other clauses and to switch on and off the 'Use' flags in the first column to get the SQL to generate properly for all cases. When the documentation and deployment databases are different databases there will be these small issues.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_tables_doc_02 (
        pk_spreadsheet_name       varchar (255)       default 'unknown'
       ,pk_spreadsheet_version    varchar (255)       default 'unknown'
       ,pk_row_number             integer             not null default 0
       ,pk_row_type               varchar (255)       not null default 'unknown'
       ,table_owner               varchar (255)       not null default 'unknown'
       ,table_name                varchar (4000)      not null default 'unknown'
       ,table_column_name         varchar (4000)      not null default 'unknown'
       ,table_column_number       integer             not null default 0
       ,column_data_type_deploy   varchar (255)       not null default 'unknown'
       ,column_data_type_document varchar (255)       not null default 'unknown'
       ,column_null_clause_deploy varchar (255)       not null default 'unknown'
       ,column_null_clause_document     varchar (255) not null default 'unknown'
       ,column_other_clauses      varchar (4000)      not null default 'unknown'
)
```

4.2.37.ctl_dw_count_sql_doc

The SeETL[DT] Workbook has now been updated to be able to generate the 'sql input rows counting statements' that were previously written by hand. These statements count the number of input rows in a view at the beginning of processing and are a part of the standard SeETL[DT] Audit processing.

A sample statement for such a row count is as follows.

```
-----------------------------------------------------------------------------------------------------------------------
--
--  spreadsheet name:       seetlwbmodeldemoworksheet01.xml spreadsheet version:        v3.0.02
--  spreadsheet file name:  c:\seetl\designtime\3.0.02\sourcecode\seetl\seetlmodeldemoworksheet01v3.0.02.xml
--  file name:              c:\temp\bi4allbase\target01\rct_z01_vf_sale_txn_01.sql
--  windows domain:         emsbidev01
--  windows user:           administrator
--  windows workstation:    emsbidev01
--  seetl current directory:  c:\seetl\designtime\3.0.02\sourcecode\seetl
--  windows version:        microsoft windows nt 5.2.3790 service pack 2
--  dot net version:        2.0.50727.1433
--  seetl workbench version:  seetl workbench 3.0.02 - 2010-09-01
--  timestamp:              17/06/2011 16:28:27
--  description:            description
--
-----------------------------------------------------------------------------------------------------------------------

 insert into dbo.ctl_file_cycle_control
 select 'z01_vf_sale_txn_01'  as pk_view_name
, (select coalesce(max(ctl_file_cycle_control.pk_file_cycle_number),0) + 1
from  dbo.ctl_file_cycle_control ctl_file_cycle_control
where 1=1 and ctl_file_cycle_control.pk_view_name = 'z01_vf_sale_txn_01'  ) as pk_file_cycle_number
, 'vf_sale_txn' as table_name
 ,(select ctl_batch_control.batch_date from dbo.ctl_batch_control ctl_batch_control where ctl_batch_control.batch_complete_flag = 0 ) as batch_date
 ,(select ctl_batch_control.pk_batch_number from dbo.ctl_batch_control ctl_batch_control where ctl_batch_control.batch_complete_flag = 0 ) as batch_number
 ,(select count(*) from dbo.z01_vf_sale_txn_01) as number_of_records

 ;
```

As can be seen from the statement. When it executes it will insert a row into the ctl_file_cycle_control table. The row will include the following elements.

1. The name of the mapping that the count statement is correlated to. Therefore the audit of the actual executed mapping statement should process the same number of rows.If it does not there is some sort of error.

2. The file cycle number. This number is the number of times this mapping has been run. A file may only be processed once per batch number. Two file cycle numbers in one batch is considered an error. A file is not necessarily processed in every batch. Batches of jobs may contain different sets of files to process.

3. The name of the underlying view or table that will be updated.

4. The batch date that the ctl_batch_control table for the current batch is saying is the batch processing date. The batch processing date can be different to the system date.

5. The batch number that the ctl_batch_control table for the current batch is saying is the batch number that is being processed.

From this information it is possible to record in the ctl_file_cycle_control table the number of rows to be processed for each input view, the file cycle in which they were processed, and the batch in which they were processed. The file cycle number and batch number are placed onto every record in the data warehouse as well as a number representing the mapping name. In prior versions the mapping name was placed onto each row. However this was seen as a little wasteful and it has been reduced to an integer. In the vast majority of cases only one mapping updates a target view. But for some processing like vm_party many mappings target and update the same underlying view or table.

4.2.37.1. ctl_dw_count_sql_doc_01

The first row of the spreadsheet defines directory that the sql statements will be written to.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
target_directory	This field contains the name of the target directory for the create count sql scripts. It is recommended that it contain no quotes and no blanks as these have not been tested and will not be supported.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_count_sql_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,target_directory       varchar (4000)    not null default 'unknown'
)
```

4.2.37.2. ctl_dw_count_sql_doc_02

The subsequent rows contain one row per sql statement to be generated. The values of the fields are self explanatory.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_row_type	**count** – SQL Row Count generation directives.
pk_target_file_name	This is the name of the file that will be created or overwritten when the generator creates the count sql. It is recommended that some form of prefix be used to identify it as the count sql for a mapping. The proposed standard prefix is rct for 'row count'.
view_owner	This is the name of the owner of the view that is to have it's row count counted.
view_name	This is the name of the view that is to have it's row count counted.
table_name	This is the name of the target table/view that the rows will be sent to when the mapping actually runs to process the rows in the input view to the mapping.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_count_sql_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_target_file_name    varchar (255)     not null default 'unknown'
      ,view_owner             varchar (255)     not null default 'unknown'
      ,view_name              varchar (255)     not null default 'unknown'
      ,table_name             varchar (255)     not null default 'unknown'
)
```

4.2.38.ctl_dw_runstats_doc

The SeETL[DT] Workbook has now been updated to be able to store runstats commands. It is well understood that this is somewhat redundant and that the run stats commands can be just as easily written directly into files. However, a large client asked us to place as much information as possible into the workbook and also place that data into the SeETL dictionary so that it was published and public knowledge.

This is seen as valuable and this is why these tabs have been added.

Because all the different databases have very different formats for the run stats commands we did not try and generate runstats commands for each database type. We simply store the runstats commands and place them into tables in the dictionary.

4.2.38.1. ctl_dw_runstats_doc_01

The first row of the spreadsheet defines directory that the sql statements will be written to.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
target_directory	This field contains the name of the target directory for the create runstats sql scripts. It is recommended that it contain no quotes and no blanks as these have not been tested and will not be supported.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_runstats_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,target_directory       varchar (4000)    not null default 'unknown'
)
```

4.2.38.2. ctl_dw_runstats_doc_02

The create run stats definitions are of the workbook contains simple fields to store the run stats sql and to abe able to query the dictionary to find the statements that have been stored without needing to query the text of the statements.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_row_type	**runstats** – Indicates an SQL Runstats generation directive.
pk_target_file_name	This is the name of the file that will be created or overwritten when the generator creates the runstats sql. It is recommended that some form of prefix be used to identify it as the runs stats sql for a line entry in this workbook. The proposed standard prefix is runstats for 'runs stats sql'.
object_owner	This is the name of the owner of the object that is to have the runstats performed against it. This is for documentation and query.
object_name	This is the name of the object that is to have the runstats performed against it. This is for documentation and query.
drop_stats_command	Some databases require statistics to be dropped or over written and require this to be specified. Therefore this command could be the drop command for the runstats or it could be a null command that does nothing depending on what the next command is.
create_stats_command	Some databases allow the rebuilding of existing stats or the creation of a new set of statistics. Depending on what commands the DBA wishes to submit to the database on a regular basis this command may be a create or rebuild stats command. Depending on what this command is the drop stats command may be different. The two commands are placed one after another in the generated file. The DBA is the authority as to exactly what should be placed into these two commands per row.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_runstats_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_target_file_name    varchar (255)     not null default 'unknown'
      ,object_owner           varchar (255)     not null default 'unknown'
      ,object_name            varchar (255)     not null default 'unknown'
      ,drop_stats_command     varchar (4000)    not null default 'unknown'
      ,create_stats_command   varchar (4000)    not null default 'unknown'
)
```

4.2.39.ctl_dw_ctas_doc

The SeETL[DT] Workbook has now been updated to be able to store create table as select commands. This sheet does not allow the creation of complex select statements. Should the create table as select be desired to be a complex statement then it is recommended that the client create a view in 'other views' to create the complex select statement and then create the create table as select based on that view.

Because all the different databases have different formats for the create table as select commands we did not try and generate create table as select commands for each database type. We simply store the create table as select commands and place them into tables in the dictionary.

4.2.39.1. ctl_dw_ctas_doc_01

The first row of the spreadsheet defines directory that the sql statements will be written to.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
target_directory	This field contains the name of the target directory for the create table as sql scripts. It is recommended that it contain no quotes and no blanks as these have not been tested and will not be supported.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_ctas_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,target_directory       varchar (4000)    not null default 'unknown'
)
```

4.2.39.2. ctl_dw_ctas_doc_02

The subsequent rows contain one row for each create table as statement that is desired. The contents of the fields is self explanatory.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_row_type	**ctas** – Indicates a create table as select generation directive.
pk_target_file_name	This is the name of the file that will be created or overwritten when the generator creates the create table as select sql. It is recommended that some form of prefix be used to identify it as the create table as select sql for a line entry in this workbook. The proposed standard prefix is ctas for 'create table as select sql'.
table_owner	This is the name of the owner of the table that is to be the newly created table as a result of the create table as select statement being executed. This is for documentation and query.
table_name	This is the name of the table that is to be the newly created table as a result of the create table as select statement being executed. This is for documentation and query.
drop_command	The DBA will have the option to either truncate the existing table having previously created it some other way or to drop the pre-existing table. Some DBAs have a great preference for truncate table over drop table. This is because in most databases the drop table also drops indexes and any statistics that are associated with the data in the table. Though, to be fair, some databases are allowing these other objects to be retained even when a table is dropped.
create_command	Depending on whether the DBA has decided to allow drop tables or truncate tables for these tables the command that is then executed is the create table command or the insert rows into table command.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_ctas_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_target_file_name    varchar (255)     not null default 'unknown'
      ,table_owner            varchar (255)     not null default 'unknown'
      ,table_name             varchar (255)     not null default 'unknown'
      ,drop_command           varchar (4000)    not null default 'unknown'
      ,create_command         varchar (4000)    not null default 'unknown'
)
```

4.2.40.ctl_dw_create_index_doc

The SeETL[DT] Workbook has now been updated to be able to store create index commands.

Because all the different databases have different formats for the create index commands we did not try and generate create index commands for each database type. We simply store the create index commands and place them into tables in the dictionary.

4.2.40.1. ctl_dw_create_index_doc_01

The first row of the spreadsheet defines directory that the sql statements will be written to.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
target_directory	This field contains the name of the target directory for the create index sql scripts. It is recommended that it contain no quotes and no blanks as these have not been tested and will not be supported.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_create_index_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,target_directory       varchar (4000)    not null default 'unknown'
)
```

4.2.40.2. ctl_dw_create_index_doc_02

The create index definitions of the workbook contains simple fields to store the create index sql and to abe able to query the dictionary to find the statements that have been stored without needing to query the text of the statements.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_row_type	**index** – Indicates a create index generation directive.
pk_target_file_name	This is the name of the file that will be created or overwritten when the generator creates the create index sql. It is recommended that some form of prefix be used to identify it as the create index sql for a line entry in this workbook. The proposed standard prefix is index for 'create index sql'.
table_owner	This is the name of the owner of the table that is to be indexed. This is for documentation and query.
table_name	This is the name of the table that is to be indexed. This is for documentation and query.
drop_command	In most databases indexes are dropped and recreatead rather than rebuilt. It is expected that this cell will contain drop index commands.
create_command	In most databases indexes are dropped and recreatead rather than rebuilt. It is expected that this cell will contain create index commands.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_create_index_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_target_file_name    varchar (255)     not null default 'unknown'
      ,table_owner            varchar (255)     not null default 'unknown'
      ,table_name             varchar (255)     not null default 'unknown'
      ,drop_command           varchar (4000)    not null default 'unknown'
      ,create_command         varchar (4000)    not null default 'unknown'
)
```

4.2.41.ctl_dw_create_storage_doc

The SeETL[DT] Workbook has now been updated to be able to store create storage objects commands.

A storage object are thinkgs such as tablespaces in DB2 and Oracle and File Groups in SQL Server. Most databases have some form of Storage Object that is required to be created to place tables and index data into.

Because all the different databases have different formats for the create storage objects commands we did not try and generate create storage objects commands for each database type. We simply store the create storage objects commands and place them into tables in the dictionary.

4.2.41.1. ctl_dw_create_storage_doc_01

The first row of the spreadsheet defines directory that the sql statements will be written to.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
target_directory	This field contains the name of the target directory for the create storage object sql scripts. It is recommended that it contain no quotes and no blanks as these have not been tested and will not be supported.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_create_storage_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,target_directory       varchar (4000)    not null default 'unknown'
)
```

4.2.41.2. ctl_dw_create_storage_doc_02

The create storage object definitions of the workbook contains simple fields to store the create storage object sql and to abe able to query the dictionary to find the statements that have been stored without needing to query the text of the statements.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_row_type	**storage** – Indicates a create storage objects generation directive.
pk_target_file_name	This is the name of the file that will be created or overwritten when the generator creates the create storage objects sql. It is recommended that some form of prefix be used to identify it as the create storage objects sql for a line entry in this workbook. The proposed standard prefix is stg for 'create storage objects sql'.
object_owner	Storage Objects are most commonly used to store tables and indexes. Hence, to be able to find the commands used we have chosen to link the create storage object command to the table that it is most closely associated with. This is the name of the owner of the table that is to be indexed. This is for documentation and query.
object_name	This is the name of the table that is to have a storage object created for it. This is for documentation and query.
drop_command	In most databases storage objects are dropped and recreatead rather than adjusted. It is expected that this cell will contain drop storage objects commands.
create_command	In most databases storage objects are dropped and recreatead rather than adjusted. It is expected that this cell will contain create storage objects commands.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_create_storage_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_target_file_name    varchar (255)     not null default 'unknown'
      ,object_owner           varchar (255)     not null default 'unknown'
      ,object_name            varchar (255)     not null default 'unknown'
      ,drop_command           varchar (4000)    not null default 'unknown'
      ,create_command         varchar (4000)    not null default 'unknown'
)
```

4.2.42.ctl_dw_permissions_doc

The SeETL[DT] Workbook has now been updated to be able to store create access permission commands.

All databases have a variety of commands to grant various access permissions. All database managers have implemented robust security models. Some are more robust and intricate than others but all databases can protect the data held within it quite securely.

Since these permissions need to be explicitly created and submitted against the database it makes sense to store them and to publish them in reports outside the database itself. In this way the security models implemented in the databases can be made transparent and clear.

Because all the different databases have different formats for the create access permissions commands we did not try and generate access permissions commands for each database type. We simply store the access permissions commands and place them into tables in the dictionary.

4.2.42.1. ctl_dw_permissions_doc_01

The first row of the spreadsheet defines directory that the sql statements will be written to.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
target_directory	This field contains the name of the target directory for the create permissions sql scripts. It is recommended that it contain no quotes and no blanks as these have not been tested and will not be supported.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_permissions_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,target_directory       varchar (4000)    not null default 'unknown'
)
```

4.2.42.2. ctl_dw_permissions_doc_02

The subsequent rows contain one row for each create permissions statement that is desired. The contents of the fields is self explanatory.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_row_type	**access** – Indicates a create access permissions generation directive.
pk_target_file_name	This is the name of the file that will be created or overwritten when the generator creates the create access permissions sql. It is recommended that some form of prefix be used to identify it as the create access permissions sql for a line entry in this workbook. The proposed standard prefix is accs for 'create access permissions sql'.
object_owner	Access Permissions are most usually granted to a group or to a table or set of tables via an owner name. It is, therefore, not a simple matter to define some fields that give good query capability to determine how to find the access permission statements in the dictionary. We have decided to stay with table owner and table name for the query fields for access permissions as this is the primary type of object to which permissions are granted. This is the name of the owner of the table that access is to be granted to or some proxy for said table. This is for documentation and query.
object_name	This is the name of the table that access is to be granted to or some proxy for said table. This is for documentation and query.
drop_command	In most databases access permissions are dropped and recreatead rather than adjusted. It is expected that this cell will contain drop access permissions commands.
create_command	In most access permissions are dropped and recreatead rather than adjusted. It is expected that this cell will contain create access permissions commands.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_permissions_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_target_file_name    varchar (255)     not null default 'unknown'
      ,object_owner           varchar (255)     not null default 'unknown'
      ,object_name            varchar (255)     not null default 'unknown'
      ,drop_command           varchar (4000)    not null default 'unknown'
      ,create_command         varchar (4000)    not null default 'unknown'
)
```

4.2.43.ctl_dw_generic_object_doc

The SeETL[DT] Workbook has now been updated to be able to store create generic objects commands.

We have attempted to upgrade the workbook to store as many objects as are commonly found in the major relational databases. However, some databases have some other objects that are unique. Further? History tells us that some new forms of objects will come out in the future.

Hence we have chosen to create a sheet for those objects that are not explicitly covered in the list of other objects supported. These are just labeled ‘generic objects’.

Because all the different databases have different formats for the create generic objects commands we did not try and generate create generic objects commands for each database type. We simply store the create generic objects commands and place them into tables in the dictionary.

4.2.43.1. ctl_dw_generic_object_doc_01

The first row of the spreadsheet defines directory that the sql statements will be written to.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
target_directory	This field contains the name of the target directory for the create generic object sql scripts. It is recommended that it contain no quotes and no blanks as these have not been tested and will not be supported.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_generic_object_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,target_directory       varchar (4000)    not null default 'unknown'
)
```

4.2.43.2. ctl_dw_generic_object_doc_02

The create generic object definitions of the workbook contains simple fields to store the create generic object sql and to abe able to query the dictionary to find the statements that have been stored without needing to query the text of the statements.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_row_type	**generic** – Indicates a create generic object generation directive.
pk_target_file_name	This is the name of the file that will be created or overwritten when the generator creates the create generic objects sql. It is recommended that some form of prefix be used to identify it as the create generic objects sql for a line entry in this workbook. The proposed standard prefix is geno for 'create generic objects sql'.
object_type	At the time of writing this manual we can have no idea of what generic object might be created. Hence we are creating a text field to store the type of object that is being created with this command so that it may be later retrieved for reporting of the various commands that exist in the dictionary to create generic objects.
object_owner	We are assuming that most objects created by such statements will have an object owner or, as a minimum, a surrogate owner. This is for documentation and query.
object_name	We are assuming that most objects created by such statements will have an object name or, as a minimum, a surrogate name that the object can be linked to for reporting purposes. This is for documentation and query.
drop_command	It is expected that most generic objects are dropped and recreatead rather than adjusted. It is expected that this cell will contain drop generic objects commands.
create_command	It is expected that most generic objects are dropped and recreatead rather than adjusted. It is expected that this cell will contain create generic objects commands.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_generic_object_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_target_file_name    varchar (255)     not null default 'unknown'
      ,object_type            varchar (255)     not null default 'unknown'
      ,object_owner           varchar (255)     not null default 'unknown'
      ,object_name            varchar (255)     not null default 'unknown'
      ,drop_command           varchar (4000)    not null default 'unknown'
      ,create_command         varchar (4000)    not null default 'unknown'
)
```

4.2.44.ctl_dw_report_doc

The SeETL[DT] Workbook has now been updated to be able to store documentation about reports that are deployed across the enterprise.

One of the biggest issues in any large enterprise is that there are hundreds if not thousands of electronic reports that are produced. These reports are often produced on different reporting technologies. They are documented poorly, if at all, and usually only the source code used to create the report in the reporting tool is documented.

The concepts behind 'self documenting reports' that were used in www.meta5.com have not been widely adopted by the leading reporting vendors Cognos, Business Objects, MicroStrategy and Microsoft. Further, these four vendors, and you can add to them Oracle, have failed to co-operate on integration of reporting information for many years now. There seems no possibility that they will co-operate any time soon.

As a result of this there is a dire need to be able to document reports across an enterprise.

We have created a small set of dictionary tables to provide such a tool should our clients wish to use it. The idea is simply that all the reports and the data on the reports can be captured via one or more workbooks and then placed into the dictionary. Capturing this information into Excel sheets is as easy as anything else that might be done. Reports will be developed in Report Services as time permits. However, we felt it was best to get this tool out into the public domain as soon as possible even though we have not performed report development yet.

Note that reports can be documented in many languages.

4.2.44.1. ctl_dw_report_doc_01

The first row of the spreadsheet defines the number of languages that are being used to document the reports.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**nl** – Number of Languages.
number_languages	This field defines the number of languages that are being used to document the reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_report_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,number_languages       integer           not null default 0
)
```

4.2.44.2. ctl_dw_report_doc_02

The report descriptions that are documented and defined in this workbook have short and long descriptions in up to 10 languages defined for them. This set of rows defines the base information and short descriptions for the reports.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**rptsd** – Report Description Short Description
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_language_number	This is a number from 1 to to indicate the language associated with this row. 1 is defined to be English. The client must define the language meanings for the values 2 to 10.
name_short	This is a short name that is assigned to a report. Usually it is about an 8 to 12 character string that contains some indication of the system/application, purpose and number of the report. It is usual to assign some form of report short name so that the technology people can use that name for quick and simple communication of which report they are talking about.
name_long	This is a long name that is assigned to a report. This is the name by which the report is generally known to the business community. It often contains the report short name for clarity.
submodel	The ***BI4ALL Data Models*** are implemented in submodels. Very frequently reports are associated with a single sub model. This sub model gives the opportunity to link the report to the submodel should that be the actual case.
report_number	It is often the case that a report is also given a number. Should the client choose to give reports numbers as well as short names that number can be stored in this cell. It is considered a string field even if it contains purely numeric characters.
frequency	This is the typical frequency with which the report will be executed.
owner	This is the Business Owner of the report. It is stored as a text field.
tech_support	This is the person or department that is responsible for maintaining this report as 'Tech Support'.
reporting_tool	This is a text field that describeds the tool that was used to create this report. Examples are "Gognos", "ReportServices", "BusinessObjects" etc.
report_sdesc	This is the description of the Report in the Nth language.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_report_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_language_number     integer           not null default 0
      ,name_short             varchar (255)     not null default 'unknown'
      ,name_long              varchar (255)     not null default 'unknown'
      ,submodel               varchar (255)     not null default 'unknown'
      ,report_number          varchar (255)     not null default 'unknown'
      ,frequency              varchar (255)     not null default 'unknown'
      ,owner                  varchar (255)     not null default 'unknown'
      ,tech_support           varchar (255)     not null default 'unknown'
      ,reporting_tool         varchar (255)     not null default 'unknown'
      ,report_sdesc           varchar (4000)    not null default 'unknown'
)
```

4.2.44.3. ctl_dw_report_doc_03

The Reports that are documented and defined in this workbook have short and long descriptions in up to 10 languages defined for them. This set of rows defines the long descriptions for the Reports.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**rptId** – Report Description Long Description
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_language_number	This is a number from 1 to to indicate the language associated with this row. 1 is defined to be English. The client must define the language meanings for the values 2 to 10.
name_short	This is a short name that is assigned to a report. Usually it is about an 8 to 12 character string that contains some indication of the system/application, purpose and number of the report. It is usual to assign some form of report short name so that the technology people can use that name for quick and simple communication of which report they are talking about.
name_long	This is a long name that is assigned to a report. This is the name by which the report is generally known to the business community. It often contains the report short name for clarity.
report_ldesc	This is the long description of the Report in the Nth language.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_report_doc_03 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_language_number     integer           not null default 0
      ,name_short             varchar (255)     not null default 'unknown'
      ,name_long              varchar (255)     not null default 'unknown'
      ,report_ldesc           varchar (4000)    not null default 'unknown'
)
```

4.2.44.4. ctl_dw_report_doc_04

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**rptip** – Report Input Parameters.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_language_number	This is a number from 1 to to indicate the language associated with this row. 1 is defined to be English. The client must define the language meanings for the values 2 to 10.
row_type	This defines the type of the row that the SeETL[DI] software will interpret the row as. There are some 'helper' values in here to make the code a little easier to write. The valid values are: • **b** – Begin. Each report should have a 'b' on the first line to warn the generator that it is the beginning of a new report definition. • **m** – Move. This is the 'normal' or default value of a row type. • **e** – End. Each report is required to have an 'end' line. This is just to be consistent with all the other worksheets like this.
name_short	This is a short name that is assigned to a report. Usually it is about an 8 to 12 character string that contains some indication of the system/application, purpose and number of the report. It is usual to assign some form of report short name so that the technology people can use that name for quick and simple communication of which report they are talking about.
name_long	This is a long name that is assigned to a report. This is the name by which the report is generally known to the business community. It often contains the report short name for clarity.
input_parameter_name	This is the name of the parameter that is being used to filter the report. It may be the name on the screen/report that is visible to the user. It might just be a column name that is used that is not visible to the user.
input_parameter_data_type	This is documentation of the Parameter Column Data Type. It explains to the user the data tpe of the parameter. In most cases parameters are strings. On rare occasions they may be numerics. In some cases dates are used and can be selected from calendars that pop up to constrain the reports. It all depends on how the reporting tools will be implemented for the client.
input_parameter_sdesc	This is the short help text of the parameter column in language N.
input_parameter_ldesc	This is the long help text of the parameter column in language N.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_report_doc_04 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_language_number     integer           not null default 0
      ,row_type               varchar (255)     not null default 'unknown'
      ,name_short             varchar (255)     not null default 'unknown'
      ,name_long              varchar (255)     not null default 'unknown'
      ,input_parameter_name   varchar (255)     not null default 'unknown'
      ,input_parameter_data_type    varchar (255)     not null default 'unknown'
      ,input_parameter_sdesc  varchar (4000)    not null default 'unknown'
      ,input_parameter_ldesc  varchar (4000)    not null default 'unknown'
)
```

4.2.44.5. ctl_dw_report_doc_05

To be able to properly document reports we must document all output fields. The most important part of the report documentation is the attempt to be able to link the field on the report all the way back to the source field that the report field came from. And this attempt includes even being able to perform that linkage for more complex fields and derivations.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**rptof** – Report Output Field.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_language_number	This is a number from 1 to to indicate the language associated with this row. 1 is defined to be English. The client must define the language meanings for the values 2 to 10.
row_type	This defines the type of the row that the SeETL[DT] software will interpret the row as. There are some 'helper' values in here to make the code a little easier to write. The valid values are: • **b** – Begin. Each report should have a 'b' on the first line to warn the generator that it is the beginning of a new report. • **m** – Move. This is the 'normal' or default value of a row type. • **e** – End. Each report is required to have an 'end' line to tell the generator this is the end of the report. This is done to make it consistent with other worksheets.
name_short	This is a short name that is assigned to a report. Usually it is about an 8 to 12 character string that contains some indication of the system/application, purpose and number of the report. It is usual to assign some form of report short name so that the technology people can use that name for quick and simple communication of which report they are talking about.
name_long	This is a long name that is assigned to a report. This is the name by which the report is generally known to the business community. It often contains the report short name for clarity.
output_column_name	This is the name of the output field that is placed onto the report. It may be something like the field name from a stored procedure or the object name in a business objects universe. It must be remembered that this is documentation only and that what should be placed into this cell is the name that is most beneficial to the client.
output_column_data_type	This is documentation of the Output Column Data Type. It explains to the user the data tpe of the output field. In most cases output columns are strings on reports. They may be able to be converted to things like dates and numerics by an 'export to' function. For example many reporting tools can export reports to Microsofts Excel.
data_model_table	If the field is the direct placement of a field from the data base onto the report then the data model table is named in this cell.
data_model_field	If the field is the direct placement of a field from the data base onto the report then the data model field is named in this cell.
output_column_sdesc	This is the short help text of the Output column in language N.
output_column_ldesc	This is the long help text of the Output column in language N.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_report_doc_05 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_language_number     integer           not null default 0
      ,row_type               varchar (255)     not null default 'unknown'
      ,name_short             varchar (255)     not null default 'unknown'
      ,name_long              varchar (255)     not null default 'unknown'
      ,output_column_name     varchar (255)     not null default 'unknown'
      ,output_column_data_type varchar (255)    not null default 'unknown'
      ,data_model_table       varchar (255)     not null default 'unknown'
      ,data_model_field       varchar (255)     not null default 'unknown'
      ,output_column_sdesc    varchar (4000)    not null default 'unknown'
      ,output_column_ldesc    varchar (4000)    not null default 'unknown'
)
```

4.2.45.ctl_dw_project_plan_doc

The SeETL[DT] Workbook has now been updated to be able to store project plans for implementation projects that can be linked to the forthcoming IBI Methodology, BI4ALL Data Models and SeETL tools.

All development projects require some form of Project Plan to be completed. MS Project has proven to be a very poor tool to use over a very long period. Most clients have some form of Project Management tool that they use and that they ask us to also use. Because of this there has been little incentive for us to build any sort of Project Planning tools.

However. Now seems to be an appropriate time to define an IBI Project Plan and to build a relatively simple project planning tool.

In the early releases this will be little more than the ability to store the Project Plan in a spreadsheet and load it up into the dictionary for reporting purposes. The reports will continue to be built in MS Report Services. Other tools may be used for reporting as well.

4.2.45.1. ctl_dw_project_plan_doc_01

The first row of the spreadsheet defines the number of Persons allocated to the project.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**np** – Number of Persons.
number_persons	This field defines the number of persons that will be assigned to the project.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_project_plan_doc_01 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,number_persons         integer           not null default 0
)
```

4.2.45.2. ctl_dw_project_plan_doc_02

The project plan must link the Persons and the Projects. This first set of fields defines the Persons assigned to the Project.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**patp** – Person Assigned to Project Description
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_person_key	This is a string key that will be used to link the Person to other rows in the dictionary.
person_initials	These are the initials of the person to be used on reports.
person_name	This is the name of the person to be used on reports.
start_date	This is the start date of the persons availability for the project. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
end_date	This is the end date of the persons availability for the project. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
percentage_available	This is the percentage of time in the available period that the person is available. It is usually availability on a weekly basis that is measured on this field but could be weekly or monthly.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_project_plan_doc_02 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_person_key          varchar (255)     not null default 'unknown'
      ,person_initials        varchar (255)     not null default 'unknown'
      ,person_name            varchar (255)     not null default 'unknown'
      ,start_date             datetime
      ,end_date               datetime
      ,percentage_available   integer           not null default 0
)
```

4.2.45.3. ctl_dw_project_plan_doc_03

The Persons working on the Project work in various roles. These are described in this section.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**pr** – Project Role Description
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_role_key	This is a string key that will be used to link the Project Role to other rows in the dictionary.
role_name	This is the name of the role that will appear on reports.
role_sdesc	This is the short description of the Project Role.
role_ldesc	This is the long description of the Project Role.
start_date	This is the start date of the role on the project. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
end_date	This is the end date of the role on the project. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
percentage_time_reqd	This is the percentage of time in the role is required to be worked for this project. Some roles are part time such as DBA resource. Some are full time but split between people.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_project_plan_doc_03 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_type            varchar (255)     not null default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_role_key            varchar (255)     not null default 'unknown'
      ,role_name              varchar (255)     not null default 'unknown'
      ,role_sdesc             varchar (255)     not null default 'unknown'
      ,role_ldesc             varchar (255)     not null default 'unknown'
      ,start_date             datetime
      ,end_date               datetime
      ,percentage_time_reqd   integer           not null default 0
)
```

4.2.45.4. ctl_dw_project_plan_doc_04

The Project Plan is made up of tasks. However these tasks can be grouped up into Project Task Groups so that the groups can be managed and closed off as a group.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**ptg** – Project Task Group Description.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_group_number	This is the Project Task Group Number which will be used as a string key to link it to other rows in the dictionary.
group_number_string	This is the Project Task Group Number which will be used as a string key to link it to other rows in the dictionary. It is intended as a string that can contain periods in the number so that the number is more easily read. It is a reformatting of the Group Number.
group_name	This is the Project Task Group Name. It is a descriptive field that will be used on reports.
plan_start_date	This is the planned start date of the Project Task Group. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
plan_end_date	This is the planned end date of the Project Task Group. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
actual_start_date	This is the actual start date of the Project Task Group. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
actual_end_date	This is the actual end date of the Project Task Group. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
group_status	This is the current status of the Project Task Group. It is a descriptive field that will be used on reports.
percentage_complete	This is the current Project Task Group Percentage Complete. It is used to track progress and completion of Project Task Groups.
group_owner_person_key	This is the link to the Person Assigned to Project Table to retrieve the details of the Person who is the owner of this Project Task Group.
group_owner_initials	These are the initials of the Person who is the Task Group Owner for easier identification on the workbook. It is redundantly stored to make it easier to know who the Group Owner Person Key links to.
group_effort_in_hours	This is the number of hours that it is expected to take for the Task Group to be completely.
group_level	Task Groups can be made into Hierarchies to more effectively manage the work that needs to be done. The Group Level field is an integer field that allows the group to participate in a Hierarchy. It is numeric.
group_signatories	Some Task Groups will need to be signed off by signatories. This field contains the initials of the signatories to the task group should the task group require signatories. If no signatories are required then “NA” is the recommended value.
group_ldesc	This is space for a detailed description of the Task Group. It is for reporting purposes only.
group_link_key	This is a string field to enable the Task Group Row to be linked to some other table via this field. It is user definable and the client is free to place any string key into this field he chooses.
estimated_internal_cost	This is the estimated cost in the clients currency units that it is estimated this task group is going to consume. It is a whole number.
estimated_external_revenue	This is the estimated external revenue in the clients currency units that will be generated for this task group if the client is something like a consulting firm that is performing tasks for their client. It is a whole number.
actual_internal_cost	This is the actual cost in the clients currency units that were consumed for the performance of this task group. It is a whole number.
actual_external_revenue	This is the actual external revenue in the clients currency units that was generated for this task group if the client is something like a consulting firm that is performing

	tasks for their client. It is a whole number.
invoiced_flag	This is a flag that can be set to “Y” or “N” to indicate whether this Task Group has been invoiced to the end client.
paid_flag	This is a flag that can be set to “Y” or “N” to indicate whether this Task Group has been invoiced to the end client and the end client has paid this amount to the bank of the client.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_project_plan_doc_04 (
        pk_spreadsheet_name       varchar (255)       default 'unknown'
       ,pk_spreadsheet_version    varchar (255)       default 'unknown'
       ,pk_row_type               varchar (255)       not null default 'unknown'
       ,pk_row_number             integer             not null default 0
       ,pk_group_number           varchar (255)       not null default 'unknown'
       ,group_number_string       varchar (255)       not null default 'unknown'
       ,group_name                varchar (255)       not null default 'unknown'
       ,plan_start_date           datetime
       ,plan_end_date             datetime
       ,actual_start_date         datetime
       ,actual_end_date           datetime
       ,group_status              varchar (255)       not null default 'unknown'
       ,percentage_complete       integer             not null default 0
       ,group_owner_person_key    varchar (255)       not null default 'unknown'
       ,group_owner_initials      varchar (255)       not null default 'unknown'
       ,group_effort_in_hours     integer             not null default 0
       ,group_level               integer             not null default 0
       ,group_signatories         varchar (255)       not null default 'unknown'
       ,group_ldesc               varchar (4000)      not null default 'unknown'
       ,group_link_key            varchar (4000)      not null default 'unknown'
       ,estimated_internal_cost   integer             not null default 0
       ,estimated_external_revenue      integer       not null default 0
       ,actual_internal_cost      integer             not null default 0
       ,actual_external_revenue   integer             not null default 0
       ,invoiced_flag             varchar (1)         not null default 'u'
       ,paid_flag                 varchar (1)         not null default 'u'
)
```

4.2.45.5. ctl_dw_project_plan_doc_05

The Project Plan is made up of tasks. This section documents each task that will be managed at the task level for the project.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**pt** – Project Task Description.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_task_number	This is the Project Task Number which will be used as a string key to link it to other rows in the dictionary.
task_number_string	This is the Project Task Number which will be used as a string key to link it to other rows in the dictionary. It is intended as a string that can contain periods in the number so that the number is more easily read. It is a reformatting of the Task Number.
group_number	This is the Project Task Group Number which will be used as a string key to link it to other rows in the dictionary.
task_name	This is the Project Task Name. It is a descriptive field that will be used on reports.
plan_start_date	This is the planned start date of the Project Task. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
plan_end_date	This is the planned end date of the Project Task. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
actual_start_date	This is the actual start date of the Project Task. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
actual_end_date	This is the actual end date of the Project Task. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
task_status	This is the current status of the Project Task. It is a descriptive field that will be used on reports.
percentage_complete	This is the current Project Task Percentage Complete. It is used to track progress and completion of Project Tasks.
task_owner_person_key	This is the link to the Person Assigned to Project Table to retrieve the details of the Person who is the owner of this Project Task.
task_owner_initials	These are the initials of the Person who is the Task Owner for easier identification on the workbook. It is redundantly stored to make it easier to know who the Task Owner Person Key links to.
task_effort_in_hours	This is the number of hours that it is expected to take for the Task to be completely.
deliverable_flag	This is a flag that can be set to “Y” or “N” to indicate whether this Task is expected to produce a specific deliverable.
deliverable_number	If this task has a deliverable to be produced as a result of performing the task then this is the deliverable number that will be linked to the Deliverables section of the workbook below.
deliverable_name	If this task has a deliverable to be produced as a result of performing the task then this is the deliverable name that will be linked to the Deliverables section of the workbook below.
deliverable_sign_off_flag	This is a flag that can be set to “Y” or “N” to indicate whether this Task is expected to produce a specific deliverable and whether that deliverable is required to be signed off by specific people.
deliverable_signatories	This is a flag that can be set to “Y” or “N” to indicate whether this Task is expected to produce a specific deliverable and whether that deliverable is required to be signed off by specific people.
task_ldesc	This is space for a detailed description of the Task. It is for reporting purposes only.
task_link_key	This is a string field to enable the Task Row to be linked to some other table via this field. It is user definable and the client is free to place any string key into this field he chooses.
estimated_internal_cost	This is the estimated cost in the clients currency units that it is estimated this task

	is going to consume. It is a whole number.
estimated_external_revenue	This is the estimated external revenue in the clients currency units that will be generated for this task if the client is something like a consulting firm that is performing tasks for their client. It is a whole number.
actual_internal_cost	This is the actual cost in the clients currency units that were consumed for the performance of this task. It is a whole number.
actual_external_revenue	This is the actual external revenue in the clients currency units that was generated for this task if the client is something like a consulting firm that is performing tasks for their client. It is a whole number.
invoiced_flag	This is a flag that can be set to “Y” or “N” to indicate whether this Task has been invoiced to the end client.
paid_flag	This is a flag that can be set to “Y” or “N” to indicate whether this Task has been invoiced to the end client and the end client has paid this amount to the bank of the client.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_project_plan_doc_05 (
       pk_spreadsheet_name       varchar (255)             default 'unknown'
      ,pk_spreadsheet_version    varchar (255)             default 'unknown'
      ,pk_row_type               varchar (255)             not null default 'unknown'
      ,pk_row_number             integer                   not null default 0
      ,pk_task_number            varchar (255)             not null default 'unknown'
      ,task_number_string        varchar (255)             not null default 'unknown'
      ,group_number              varchar (255)             not null default 'unknown'
      ,task_name                 varchar (255)             not null default 'unknown'
      ,plan_start_date           datetime
      ,plan_end_date             datetime
      ,actual_start_date         datetime
      ,actual_end_date           datetime
      ,task_status               varchar (255)             not null default 'unknown'
      ,percentage_complete       integer                   not null default 0
      ,task_owner_person_key     varchar (255)             not null default 'unknown'
      ,task_owner_initials       varchar (255)             not null default 'unknown'
      ,task_effort_in_hours      integer                   not null default 0
      ,deliverable_flag          varchar (1)               not null default 'u'
      ,deliverable_number        varchar (255)             not null default 'unknown'
      ,deliverable_name          varchar (255)             not null default 'unknown'
      ,deliverable_sign_off_flag varchar (1)               not null default 'u'
      ,deliverable_signatories   varchar (255)             not null default 'unknown'
      ,task_ldesc                varchar (255)             not null default 'unknown'
      ,task_link_key             varchar (255)             not null default 'unknown'
      ,estimated_internal_cost   integer                   not null default 0
      ,estimated_external_revenue      integer             not null default 0
      ,actual_internal_cost      integer                   not null default 0
      ,actual_external_revenue   integer                   not null default 0
      ,invoiced_flag             varchar (1)               not null default 'u'
      ,paid_flag                 varchar (1)               not null default 'u'
)
```

4.2.45.6. ctl_dw_project_plan_doc_06

In prior sheet the Persons have been defined and the Tasks have been defined. Now we need to link the Persons to the Tasks. There may be many Persons assigned to one Task and many tasks may be assigned to one Person. There is an obvious many to many relationship between Persons and Tasks.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**patt** – Person Assigned To Task Description.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_person_key	This is a string key that will be used to link the Person to other rows in the dictionary.
pk_role_key	This is a string key that will be used to link the Project Role to other rows in the dictionary.
person_initials	These are the initials of the person to be used on reports.
person_name	This is the name of the person to be used on reports.
task_number	This is the Project Task Number which will be used as a string key to link it to other rows in the dictionary.
task_role	This is the short description of the Project Role.
actual_start_date	This is the start date of the Person on the Task. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
actual_end_date	This is the end date of the Person on the Task. It is required to be in Quotes and is in the format “dd/mm/yyyy”.
percentage_time_reqd	This is the percentage of time in the Person is assigned to the Role for this task. It is a whole number.
assignment_notes	This is a free format text field for documentation purposes should it be needed.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_project_plan_doc_06 (
        pk_spreadsheet_name       varchar (255)             default 'unknown'
       ,pk_spreadsheet_version    varchar (255)             default 'unknown'
       ,pk_row_type               varchar (255)             not null default 'unknown'
       ,pk_row_number             integer                   not null default 0
       ,pk_person_key             varchar (255)             not null default 'unknown'
       ,pk_role_key               varchar (255)             not null default 'unknown'
       ,person_initials           varchar (255)             not null default 'unknown'
       ,person_name               varchar (255)             not null default 'unknown'
       ,task_number               varchar (255)             not null default 'unknown'
       ,task_role                 varchar (255)             not null default 'unknown'
       ,actual_start_date         datetime
       ,actual_end_date           datetime
       ,percentage_time_reqd      integer                   not null default 0
       ,assignment_notes          varchar (4000)            not null default 'unknown'
)
```

4.2.45.7. ctl_dw_project_plan_doc_07

Tasks very frequently have sets of dependencies. This tool is not a project management tool in that it does not have an active engine to build project plans based on dependencies. There are a multitude of such project planning tools out there. The trouble is that none of them do a very good job of highly iterative projects and they are more trouble than they are worth in that area. The task dependencies in this sheet are for documentation and communication purposes only. The dates and other items are required to be maintained by hand in this simple project management tool.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**td** – Task Dependency Description.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_task_pre_dependency	This is the string key of the Task that is the “pre” or left hand side dependency. Since the dependency may be ‘co-dependency’ it is not strictly true that this is always a pre-dependency.
pk_task_pre_dependency_ sdesc	This is the name of the task that is the “pre” or left hand side dependency. It is redundantly included so as to make the workbook more meaningful to read.
pk_task_post_dependency	This is the string key of the Task that is the “post” or right hand side dependency. Since the dependency may be ‘co-dependency’ it is not strictly true that this is always a post-dependency.
pk_task_post_dependency_ sdesc	This is the name of the task that is the “post” or right hand side dependency. It is redundantly included so as to make the workbook more meaningful to read.
dependency_type	This is the type of the dependency. It might be such things as ‘Start-to-Finish’ or ‘Start-to-Start’ etc. It is documentation and not used in any processing.
lag_in_days	This is the number of days lag between the pre and post dependent tasks.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_project_plan_doc_07 (
        pk_spreadsheet_name             varchar (255)       default 'unknown'
       ,pk_spreadsheet_version          varchar (255)       default 'unknown'
       ,pk_row_type                     varchar (255)       not null default 'unknown'
       ,pk_row_number                   integer             not null default 0
       ,pk_task_pre_dependency          varchar (255)       not null default 'unknown'
       ,pk_task_pre_dependency_sdesc    varchar (255)       not null default 'unknown'
       ,pk_task_post_dependency         varchar (255)       not null default 'unknown'
       ,pk_task_post_dependency_sdesc   varchar (255)       not null default 'unknown'
       ,dependency_type                 varchar (255)       not null default 'unknown'
       ,lag_in_days                     integer             not null default 0
)
```

4.2.45.8. ctl_dw_project_plan_doc_08

During Projects a number of Deliverables will be handed over to the client. Some of these deliverables go into production at the end clients site. The client may choose to document deliverables at any level he wishes to. It might not make sense to document deliverables at the level of sql file produced or even at mapping or report level.

There will be tasks that deliver reports etc and the client may decide the fact that the report is delivered and also documented in the Reports Documentation Workbook might be enough documentation on the report. The client may decide that there is little need to also document the deliverable of the report in this section of the worksheet.

The usage of the worksheet and level of detail for deliverables is to be decided by the client project manager.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_type	**pd** – Project Deliverable Description.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_deliberable_number	This is the string key for the deliverable. It can be linked to tasks.
deliverable_name	This is the name of the deliverable.
deliverable_sign_off_flag	This is a flag that can be set to “Y” or “N” to indicate whether this Deliverable is required to be signed off.
deliverable_signatories	Some Deliverables will need to be signed off by signatories. This field contains the initials of the signatories to the Deliverable should the task require signatories. If no signatories are required then “NA” is the recommended value.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_project_plan_doc_08 (
        pk_spreadsheet_name       varchar (255)             default 'unknown'
       ,pk_spreadsheet_version    varchar (255)             default 'unknown'
       ,pk_row_type               varchar (255)             not null default 'unknown'
       ,pk_row_number             integer                   not null default 0
       ,pk_deliberable_number     varchar (255)             not null default 'unknown'
       ,deliverable_name          varchar (255)             not null default 'unknown'
       ,deliverable_sign_off_flag varchar (1)               not null default 'u'
       ,deliverable_signatories   varchar (255)             not null default 'unknown'
)
```

4.2.46.ctl_dw_biz_reqts_doc

The SeETL[DT] Workbook has been upgraded to store documentation of Business Requirements for any particular project though in this case the emphasis is on Business Intelligence Projects. The Business Requirements entered are made available to all users via a suite of reports.

4.2.46.1. ctl_dw_biz_reqts_doc_01

The first table stores the Business Requirements workbook.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_biz_reqt_number	This is the Business Requirement Number that will be used to identify the Business Requirement on reports and in conversations.
biz_reqt_name	This is the Business Requirement Name that will be used to identify the Business Requirement on reports and in conversations.
biz_reqt_sdesc	This is the Business Requirement Short Description that will be used to describe the Business Requirement on reports and in conversations.
biz_reqt_ldesc	This is the Business Requirement Long Description that will be used to describe the Business Requirement on reports and in conversations.
owner_name	This is the Business Requirement Owner Name that will be used to document the Business Requirement on reports.
owner_department	This is the Business Requirement Owner Department that will be used to document the Business Requirement on reports.
biz_reqt_importance	This is the Business Requirement Importance that will be used to document the Business Requirement Importance on reports.
biz_reqt_priority	This is the Business Requirement Priority that will be used to document the Business Requirement Priority on reports.
planned_iteration	This is the Business Requirement Delivered In Iteration information that will be used to document the Business Requirement Delivered In Iteration information on reports.
biz_reqt_status	This is the Business Requirement Status information that will be used to document the Business Requirement Status information on reports.
documentation_link	This is the Business Requirement Link to Documentation information that will be used to document the Business Requirement Link to Documentation information on reports.
last_review_date	This is the Business Requirement Last Review Date information that will be used to document the Business Requirement Last Review Date information on reports.
biz_reqt_notes01	This is the Business Requirement Notes1 information that will be used to document the Business Requirement Notes1 information on reports.
biz_reqt_notes02	This is the Business Requirement Notes2 information that will be used to document the Business Requirement Notes2 information on reports.
biz_reqt_notes03	This is the Business Requirement Notes3 information that will be used to document the Business Requirement Notes3 information on reports.
biz_reqt_notes04	This is the Business Requirement Notes4 information that will be used to document the Business Requirement Notes4 information on reports.
biz_reqt_notes05	This is the Business Requirement Notes5 information that will be used to document the Business Requirement Notes5 information on reports.
biz_reqt_notes06	This is the Business Requirement Notes6 information that will be used to document the Business Requirement Notes6 information on reports.
biz_reqt_notes07	This is the Business Requirement Notes7 information that will be used to document the Business Requirement Notes7 information on reports.
biz_reqt_notes08	This is the Business Requirement Notes8 information that will be used to document the Business Requirement Notes8 information on reports.
biz_reqt_notes09	This is the Business Requirement Notes9 information that will be used to

	document the Business Requirement Notes9 information on reports.
biz_reqt_notes10	This is the Business Requirement Notes10 information that will be used to document the Business Requirement Notes10 information on reports.
user_col_01	This is the Business Requirement UserCol01 information that will be used to document the Business Requirement UserCol01 information on reports.
user_col_02	This is the Business Requirement UserCol02 information that will be used to document the Business Requirement UserCol02 information on reports.
user_col_03	This is the Business Requirement UserCol03 information that will be used to document the Business Requirement UserCol03 information on reports.
user_col_04	This is the Business Requirement UserCol04 information that will be used to document the Business Requirement UserCol04 information on reports.
user_col_05	This is the Business Requirement UserCol05 information that will be used to document the Business Requirement UserCol05 information on reports.
user_col_06	This is the Business Requirement UserCol06 information that will be used to document the Business Requirement UserCol06 information on reports.
user_col_07	This is the Business Requirement UserCol07 information that will be used to document the Business Requirement UserCol07 information on reports.
user_col_08	This is the Business Requirement UserCol08 information that will be used to document the Business Requirement UserCol08 information on reports.
user_col_09	This is the Business Requirement UserCol09 information that will be used to document the Business Requirement UserCol09 information on reports.
user_col_10	This is the Business Requirement UserCol10 information that will be used to document the Business Requirement UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_01 (
       pk_spreadsheet_name    varchar (255)          default 'unknown'
      ,pk_spreadsheet_version varchar (255)          default 'unknown'
      ,pk_row_number          integer                not null default 0
      ,pk_biz_reqt_number     varchar (255           not null default 'unknown'
      ,biz_reqt_name          varchar (255)          not null default 'unknown'
      ,biz_reqt_sdesc         varchar (255)          not null default 'unknown'
      ,biz_reqt_ldesc         varchar (2000)         not null default 'unknown'
      ,owner_name             varchar (255)          not null default 'unknown'
      ,owner_department       varchar (255)          not null default 'unknown'
      ,biz_reqt_importance    varchar (255)          not null default 'unknown'
      ,biz_reqt_priority      varchar (255)          not null default 'unknown'
      ,planned_iteration      varchar (255)          not null default 'unknown'
      ,biz_reqt_status        varchar (255)          not null default 'unknown'
      ,documentation_link     varchar (255)          not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,biz_reqt_notes01       varchar (1000)         default 'unknown'
      ,biz_reqt_notes02       varchar (1000)         default 'unknown'
      ,biz_reqt_notes03       varchar (1000)         default 'unknown'
      ,biz_reqt_notes04       varchar (1000)         default 'unknown'
      ,biz_reqt_notes05       varchar (1000)         default 'unknown'
      ,biz_reqt_notes06       varchar (1000)         default 'unknown'
      ,biz_reqt_notes07       varchar (1000)         default 'unknown'
      ,biz_reqt_notes08       varchar (1000)         default 'unknown'
      ,biz_reqt_notes09       varchar (1000)         default 'unknown'
      ,biz_reqt_notes10       varchar (1000)         default 'unknown'
      ,user_col_01            varchar (1000)         default 'unknown'
      ,user_col_02            varchar (1000)         default 'unknown'
      ,user_col_03            varchar (1000)         default 'unknown'
      ,user_col_04            varchar (1000)         default 'unknown'
      ,user_col_05            varchar (1000)         default 'unknown'
      ,user_col_06            varchar (1000)         default 'unknown'
      ,user_col_07            varchar (1000)         default 'unknown'
      ,user_col_08            varchar (1000)         default 'unknown'
      ,user_col_09            varchar (1000)         default 'unknown'
      ,user_col_10            varchar (1000)         default 'unknown'
)
```

4.2.46.2. ctl_dw_biz_reqts_doc_02

The first table stores the Project Objectives workbook.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_project_objective_ number	This is the Project Objective Number that will be used to identify the Project Objective on reports and in conversations.
project_objective_name	This is the Project Objective Name that will be used to identify the Project Objective on reports and in conversations.
project_objective_sdesc	This is the Project Objective Short Description that will be used to describe the Project Objective on reports and in conversations.
project_objective_ldesc	This is the Project Objective Long Description that will be used to describe the Project Objective on reports and in conversations.
owner_name	This is the Project Objective Owner Name that will be used to document the Project Objective on reports.
owner_department	This is the Project Objective Owner Department that will be used to document the Project Objective on reports.
project_objective_ importance	This is the Project Objective Importance that will be used to document the Project Objective Importance on reports.
project_objective_priority	This is the Project Objective Priority that will be used to document the Project Objective Priority on reports.
planned_iteration	This is the Project Objective Delivered In Iteration information that will be used to document the Project Objective Delivered In Iteration information on reports.
biz_reqt_status	This is the Project Objective Status information that will be used to document the Project Objective Status information on reports.
documentation_link	This is the Project Objective Link to Documentation information that will be used to document the Project Objective Link to Documentation information on reports.
last_review_date	This is the Project Objective Last Review Date information that will be used to document the Project Objective Last Review Date information on reports.
project_objective_notes01	This is the Project Objective Notes1 information that will be used to document the Project Objective Notes1 information on reports.
project_objective_notes02	This is the Project Objective Notes2 information that will be used to document the Project Objective Notes2 information on reports.
project_objective_notes03	This is the Project Objective Notes3 information that will be used to document the Project Objective Notes3 information on reports.
project_objective_notes04	This is the Project Objective Notes4 information that will be used to document the Project Objective Notes4 information on reports.
project_objective_notes05	This is the Project Objective Notes5 information that will be used to document the Project Objective Notes5 information on reports.
project_objective_notes06	This is the Project Objective Notes6 information that will be used to document the Project Objective Notes6 information on reports.
project_objective_notes07	This is the Project Objective Notes7 information that will be used to document the Project Objective Notes7 information on reports.
project_objective_notes08	This is the Project Objective Notes8 information that will be used to document the Project Objective Notes8 information on reports.
project_objective_notes09	This is the Project Objective Notes9 information that will be used to document the Project Objective Notes9 information on reports.
project_objective_notes10	This is the Project Objective Notes10 information that will be used to document the Project Objective Notes10 information on reports.
user_col_01	This is the Project Objective UserCol01 information that will be used to document the Project Objective UserCol01 information on reports.
user_col_02	This is the Project Objective UserCol02 information that will be used to document the Project Objective UserCol02 information on reports.
user_col_03	This is the Project Objective UserCol03 information that will be used to document

	the Project Objective UserCol03 information on reports.
user_col_04	This is the Project Objective UserCol04 information that will be used to document the Project Objective UserCol04 information on reports.
user_col_05	This is the Project Objective UserCol05 information that will be used to document the Project Objective UserCol05 information on reports.
user_col_06	This is the Project Objective UserCol06 information that will be used to document the Project Objective UserCol06 information on reports.
user_col_07	This is the Project Objective UserCol07 information that will be used to document the Project Objective UserCol07 information on reports.
user_col_08	This is the Project Objective UserCol08 information that will be used to document the Project Objective UserCol08 information on reports.
user_col_09	This is the Project Objective UserCol09 information that will be used to document the Project Objective UserCol09 information on reports.
user_col_10	This is the Project Objective UserCol10 information that will be used to document the Project Objective UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_02 (
       pk_spreadsheet_name          varchar (255)     default 'unknown'
      ,pk_spreadsheet_version       varchar (255)     default 'unknown'
      ,pk_row_number                integer           not null default 0
      ,pk_project_objective_number  varchar (255)     not null default 'unknown'
      ,project_objective_name       varchar (255)     not null default 'unknown'
      ,project_objective_sdesc      varchar (255)     not null default 'unknown'
      ,project_objective_ldesc      varchar (2000)    not null default 'unknown'
      ,owner_name                   varchar (255)     not null default 'unknown'
      ,owner_department             varchar (255)     not null default 'unknown'
      ,project_objective_importance varchar (255)     not null default 'unknown'
      ,project_objective_priority   varchar (255)     not null default 'unknown'
      ,planned_iteration            varchar (255)     not null default 'unknown'
      ,project_objective_status     varchar (255)     not null default 'unknown'
      ,documentation_link           varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,project_objective_notes01    varchar (1000)    default 'unknown'
      ,project_objective_notes02    varchar (1000)    default 'unknown'
      ,project_objective_notes03    varchar (1000)    default 'unknown'
      ,project_objective_notes04    varchar (1000)    default 'unknown'
      ,project_objective_notes05    varchar (1000)    default 'unknown'
      ,project_objective_notes06    varchar (1000)    default 'unknown'
      ,project_objective_notes07    varchar (1000)    default 'unknown'
      ,project_objective_notes08    varchar (1000)    default 'unknown'
      ,project_objective_notes09    varchar (1000)    default 'unknown'
      ,project_objective_notes10    varchar (1000)    default 'unknown'
      ,user_col_01                  varchar (1000)    default 'unknown'
      ,user_col_02                  varchar (1000)    default 'unknown'
      ,user_col_03                  varchar (1000)    default 'unknown'
      ,user_col_04                  varchar (1000)    default 'unknown'
      ,user_col_05                  varchar (1000)    default 'unknown'
      ,user_col_06                  varchar (1000)    default 'unknown'
      ,user_col_07                  varchar (1000)    default 'unknown'
      ,user_col_08                  varchar (1000)    default 'unknown'
      ,user_col_09                  varchar (1000)    default 'unknown'
      ,user_col_10                  varchar (1000)    default 'unknown'
)
```

4.2.46.3. ctl_dw_biz_reqts_doc_03

The first table stores the Critical Success Factor workbook.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_csf_number	This is the Critical Success Factor Number that will be used to identify the Critical Success Factor on reports and in conversations.
csf_name	This is the Critical Success Factor Name that will be used to identify the Critical Success Factor on reports and in conversations.
csf_sdesc	This is the Critical Success Factor Short Description that will be used to describe the Critical Success Factor on reports and in conversations.
csf_ldesc	This is the Critical Success Factor Long Description that will be used to describe the Critical Success Factor on reports and in conversations.
owner_name	This is the Critical Success Factor Owner Name that will be used to document the Critical Success Factor on reports.
mitigation_stratgy	This is the Critical Success Factor mitigation strategy that will be used to document the Critical Success Factor on reports.
owner_department	This is the Critical Success Factor Owner Department that will be used to document the Critical Success Factor on reports.
csf_importance	This is the Critical Success Factor Importance that will be used to document the Critical Success Factor Importance on reports.
csf_priority	This is the Critical Success Factor Priority that will be used to document the Critical Success Factor Priority on reports.
csf_potential_impact	This is the Critical Success Factor potential impact that will be used to document the Critical Success Factor Priority on reports.
planned_iteration	This is the Critical Success Factor Delivered In Iteration information that will be used to document the Critical Success Factor Delivered In Iteration information on reports.
csf_status	This is the Critical Success Factor Status information that will be used to document the Critical Success Factor Status information on reports.
documentation_link	This is the Critical Success Factor Link to Documentation information that will be used to document the Critical Success Factor Link to Documentation information on reports.
last_review_date	This is the Critical Success Factor Last Review Date information that will be used to document the Critical Success Factor Last Review Date information on reports.
csf_notes01	This is the Critical Success Factor Notes1 information that will be used to document the Critical Success Factor Notes1 information on reports.
csf_notes02	This is the Critical Success Factor Notes2 information that will be used to document the Critical Success Factor Notes2 information on reports.
csf_notes03	This is the Critical Success Factor Notes3 information that will be used to document the Critical Success Factor Notes3 information on reports.
csf_notes04	This is the Critical Success Factor Notes4 information that will be used to document the Critical Success Factor Notes4 information on reports.
csf_notes05	This is the Critical Success Factor Notes5 information that will be used to document the Critical Success Factor Notes5 information on reports.
csf_notes06	This is the Critical Success Factor Notes6 information that will be used to document the Critical Success Factor Notes6 information on reports.
csf_notes07	This is the Critical Success Factor Notes7 information that will be used to document the Critical Success Factor Notes7 information on reports.
csf_notes08	This is the Critical Success Factor Notes8 information that will be used to document the Critical Success Factor Notes8 information on reports.
csf_notes09	This is the Critical Success Factor Notes9 information that will be used to document the Critical Success Factor Notes9 information on reports.
csf_notes10	This is the Critical Success Factor Notes10 information that will be used to

	document the Critical Success Factor Notes10 information on reports.
user_col_01	This is the Critical Success Factor UserCol01 information that will be used to document the Critical Success Factor UserCol01 information on reports.
user_col_02	This is the Critical Success Factor UserCol02 information that will be used to document the Critical Success Factor UserCol02 information on reports.
user_col_03	This is the Critical Success Factor UserCol03 information that will be used to document the Critical Success Factor UserCol03 information on reports.
user_col_04	This is the Critical Success Factor UserCol04 information that will be used to document the Critical Success Factor UserCol04 information on reports.
user_col_05	This is the Critical Success Factor UserCol05 information that will be used to document the Critical Success Factor UserCol05 information on reports.
user_col_06	This is the Critical Success Factor UserCol06 information that will be used to document the Critical Success Factor UserCol06 information on reports.
user_col_07	This is the Critical Success Factor UserCol07 information that will be used to document the Critical Success Factor UserCol07 information on reports.
user_col_08	This is the Critical Success Factor UserCol08 information that will be used to document the Critical Success Factor UserCol08 information on reports.
user_col_09	This is the Critical Success Factor UserCol09 information that will be used to document the Critical Success Factor UserCol09 information on reports.
user_col_10	This is the Critical Success Factor UserCol10 information that will be used to document the Critical Success Factor UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_03 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_csf_number          varchar (255      not null default 'unknown'
      ,csf_name               varchar (255)     not null default 'unknown'
      ,csf_sdesc              varchar (255)     not null default 'unknown'
      ,csf_ldesc              varchar (2000)    not null default 'unknown'
      ,mitigation_stratgy     varchar (1000)    not null default 'unknown'
      ,owner_name             varchar (255)     not null default 'unknown'
      ,owner_department       varchar (255)     not null default 'unknown'
      ,csf_importance         varchar (255)     not null default 'unknown'
      ,csf_priority           varchar (255)     not null default 'unknown'
      ,csf_potential_impact   varchar (1000)    not null default 'unknown'
      ,planned_iteration      varchar (255)     not null default 'unknown'
      ,csf_status             varchar (255)     not null default 'unknown'
      ,documentation_link     varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,csf_notes01            varchar (1000)    default 'unknown'
      ,csf_notes02            varchar (1000)    default 'unknown'
      ,csf_notes03            varchar (1000)    default 'unknown'
      ,csf_notes04            varchar (1000)    default 'unknown'
      ,csf_notes05            varchar (1000)    default 'unknown'
      ,csf_notes06            varchar (1000)    default 'unknown'
      ,csf_notes07            varchar (1000)    default 'unknown'
      ,csf_notes08            varchar (1000)    default 'unknown'
      ,csf_notes09            varchar (1000)    default 'unknown'
      ,csf_notes10            varchar (1000)    default 'unknown'
      ,user_col_01            varchar (1000)    default 'unknown'
      ,user_col_02            varchar (1000)    default 'unknown'
      ,user_col_03            varchar (1000)    default 'unknown'
      ,user_col_04            varchar (1000)    default 'unknown'
      ,user_col_05            varchar (1000)    default 'unknown'
      ,user_col_06            varchar (1000)    default 'unknown'
      ,user_col_07            varchar (1000)    default 'unknown'
      ,user_col_08            varchar (1000)    default 'unknown'
      ,user_col_09            varchar (1000)    default 'unknown'
      ,user_col_10            varchar (1000)    default 'unknown'
)
```

4.2.46.4. ctl_dw_biz_reqts_doc_04

The first table stores the Major Dimensions workbook.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_dimension_number	This is the Dimension Number that will be used to identify the Dimension on reports and in conversations.
dimension_name	This is the Dimension Name that will be used to identify the Dimension on reports and in conversations.
dimension_sdesc	This is the Dimension Short Description that will be used to describe the Dimension on reports and in conversations.
dimension_ldesc	This is the Dimension Long Description that will be used to describe the Dimension on reports and in conversations.
owner_name	This is the Dimension Owner Name that will be used to document the Dimension on reports.
owner_department	This is the Dimension Owner Department that will be used to document the Dimension on reports.
dimension_importance	This is the Dimension Importance that will be used to document the Dimension Importance on reports.
dimension_priority	This is the Dimension Priority that will be used to document the Dimension Priority on reports.
planned_iteration	This is the Dimension Delivered In Iteration information that will be used to document the Dimension Delivered In Iteration information on reports.
dimension_status	This is the Dimension Status information that will be used to document the Dimension Status information on reports.
documentation_link	This is the Dimension Link to Documentation information that will be used to document the Dimension Link to Documentation information on reports.
last_review_date	This is the Dimension Last Review Date information that will be used to document the Dimension Last Review Date information on reports.
dimension_notes01	This is the Dimension Notes1 information that will be used to document the Dimension Notes1 information on reports.
dimension_notes02	This is the Dimension Notes2 information that will be used to document the Dimension Notes2 information on reports.
dimension_notes03	This is the Dimension Notes3 information that will be used to document the Dimension Notes3 information on reports.
dimension_notes04	This is the Dimension Notes4 information that will be used to document the Dimension Notes4 information on reports.
dimension_notes05	This is the Dimension Notes5 information that will be used to document the Dimension Notes5 information on reports.
dimension_notes06	This is the Dimension Notes6 information that will be used to document the Dimension Notes6 information on reports.
dimension_notes07	This is the Dimension Notes7 information that will be used to document the Dimension Notes7 information on reports.
dimension_notes08	This is the Dimension Notes8 information that will be used to document the Dimension Notes8 information on reports.
dimension_notes09	This is the Dimension Notes9 information that will be used to document the Dimension Notes9 information on reports.
dimension_notes10	This is the Dimension Notes10 information that will be used to document the Dimension Notes10 information on reports.
user_col_01	This is the Dimension UserCol01 information that will be used to document the Dimension UserCol01 information on reports.
user_col_02	This is the Dimension UserCol02 information that will be used to document the Dimension UserCol02 information on reports.
user_col_03	This is the Dimension UserCol03 information that will be used to document the

	Dimension UserCol03 information on reports.
user_col_04	This is the Dimension UserCol04 information that will be used to document the Dimension UserCol04 information on reports.
user_col_05	This is the Dimension UserCol05 information that will be used to document the Dimension UserCol05 information on reports.
user_col_06	This is the Dimension UserCol06 information that will be used to document the Dimension UserCol06 information on reports.
user_col_07	This is the Dimension UserCol07 information that will be used to document the Dimension UserCol07 information on reports.
user_col_08	This is the Dimension UserCol08 information that will be used to document the Dimension UserCol08 information on reports.
user_col_09	This is the Dimension UserCol09 information that will be used to document the Dimension UserCol09 information on reports.
user_col_10	This is the Dimension UserCol10 information that will be used to document the Dimension UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_04 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_dimension_number    varchar (255)     not null default 'unknown'
      ,dimension_name         varchar (255)     not null default 'unknown'
      ,dimension_sdesc        varchar (255)     not null default 'unknown'
      ,dimension_ldesc        varchar (2000)    not null default 'unknown'
      ,owner_name             varchar (255)     not null default 'unknown'
      ,owner_department       varchar (255)     not null default 'unknown'
      ,dimension_importance   varchar (255)     not null default 'unknown'
      ,dimension_priority     varchar (255)     not null default 'unknown'
      ,planned_iteration      varchar (255)     not null default 'unknown'
      ,dimension_status       varchar (255)     not null default 'unknown'
      ,documentation_link     varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,dimension_notes01      varchar (1000)    default 'unknown'
      ,dimension_notes02      varchar (1000)    default 'unknown'
      ,dimension_notes03      varchar (1000)    default 'unknown'
      ,dimension_notes04      varchar (1000)    default 'unknown'
      ,dimension_notes05      varchar (1000)    default 'unknown'
      ,dimension_notes06      varchar (1000)    default 'unknown'
      ,dimension_notes07      varchar (1000)    default 'unknown'
      ,dimension_notes08      varchar (1000)    default 'unknown'
      ,dimension_notes09      varchar (1000)    default 'unknown'
      ,dimension_notes10      varchar (1000)    default 'unknown'
      ,user_col_01            varchar (1000)    default 'unknown'
      ,user_col_02            varchar (1000)    default 'unknown'
      ,user_col_03            varchar (1000)    default 'unknown'
      ,user_col_04            varchar (1000)    default 'unknown'
      ,user_col_05            varchar (1000)    default 'unknown'
      ,user_col_06            varchar (1000)    default 'unknown'
      ,user_col_07            varchar (1000)    default 'unknown'
      ,user_col_08            varchar (1000)    default 'unknown'
      ,user_col_09            varchar (1000)    default 'unknown'
      ,user_col_10            varchar (1000)    default 'unknown'
)
```

4.2.46.5. ctl_dw_biz_reqts_doc_05

The first table stores the Major Facts workbook.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_fact_number	This is the Fact Number that will be used to identify the Fact on reports and in conversations.
fact_name	This is the Fact Name that will be used to identify the Fact on reports and in conversations.
fact_sdesc	This is the Fact Short Description that will be used to describe the Fact on reports and in conversations.
fact_ldesc	This is the Fact Long Description that will be used to describe the Fact on reports and in conversations.
owner_name	This is the Fact Owner Name that will be used to document the Fact on reports.
owner_department	This is the Fact Owner Department that will be used to document the Fact on reports.
fact_importance	This is the Fact Importance that will be used to document the Fact Importance on reports.
fact_priority	This is the Fact Priority that will be used to document the Fact Priority on reports.
planned_iteration	This is the Fact Delivered In Iteration information that will be used to document the Fact Delivered In Iteration information on reports.
fact_status	This is the Fact Status information that will be used to document the Fact Status information on reports.
documentation_link	This is the Fact Link to Documentation information that will be used to document the Fact Link to Documentation information on reports.
last_review_date	This is the Fact Last Review Date information that will be used to document the Fact Last Review Date information on reports.
fact_notes01	This is the Fact Notes1 information that will be used to document the Fact Notes1 information on reports.
fact_notes02	This is the Fact Notes2 information that will be used to document the Fact Notes2 information on reports.
fact_notes03	This is the Fact Notes3 information that will be used to document the Fact Notes3 information on reports.
fact_notes04	This is the Fact Notes4 information that will be used to document the Fact Notes4 information on reports.
fact_notes05	This is the Fact Notes5 information that will be used to document the Fact Notes5 information on reports.
fact_notes06	This is the Fact Notes6 information that will be used to document the Fact Notes6 information on reports.
fact_notes07	This is the Fact Notes7 information that will be used to document the Fact Notes7 information on reports.
fact_notes08	This is the Fact Notes8 information that will be used to document the Fact Notes8 information on reports.
fact_notes09	This is the Fact Notes9 information that will be used to document the Fact Notes9 information on reports.
fact_notes10	This is the Fact Notes10 information that will be used to document the Fact Notes10 information on reports.
user_col_01	This is the Fact UserCol01 information that will be used to document the Fact UserCol01 information on reports.
user_col_02	This is the Fact UserCol02 information that will be used to document the Fact UserCol02 information on reports.
user_col_03	This is the Fact UserCol03 information that will be used to document the Fact UserCol03 information on reports.
user_col_04	This is the Fact UserCol04 information that will be used to document the Fact

	UserCol04 information on reports.
user_col_05	This is the Fact UserCol05 information that will be used to document the Fact UserCol05 information on reports.
user_col_06	This is the Fact UserCol06 information that will be used to document the Fact UserCol06 information on reports.
user_col_07	This is the Fact UserCol07 information that will be used to document the Fact UserCol07 information on reports.
user_col_08	This is the Fact UserCol08 information that will be used to document the Fact UserCol08 information on reports.
user_col_09	This is the Fact UserCol09 information that will be used to document the Fact UserCol09 information on reports.
user_col_10	This is the Fact UserCol10 information that will be used to document the Fact UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_05 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_fact_number         varchar (255)     not null default 'unknown'
      ,fact_name              varchar (255)     not null default 'unknown'
      ,fact_sdesc             varchar (255)     not null default 'unknown'
      ,fact_ldesc             varchar (2000)    not null default 'unknown'
      ,owner_name             varchar (255)     not null default 'unknown'
      ,owner_department       varchar (255)     not null default 'unknown'
      ,fact_importance        varchar (255)     not null default 'unknown'
      ,fact_priority          varchar (255)     not null default 'unknown'
      ,planned_iteration      varchar (255)     not null default 'unknown'
      ,fact_status            varchar (255)     not null default 'unknown'
      ,documentation_link     varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,fact_notes01           varchar (1000)    default 'unknown'
      ,fact_notes02           varchar (1000)    default 'unknown'
      ,fact_notes03           varchar (1000)    default 'unknown'
      ,fact_notes04           varchar (1000)    default 'unknown'
      ,fact_notes05           varchar (1000)    default 'unknown'
      ,fact_notes06           varchar (1000)    default 'unknown'
      ,fact_notes07           varchar (1000)    default 'unknown'
      ,fact_notes08           varchar (1000)    default 'unknown'
      ,fact_notes09           varchar (1000)    default 'unknown'
      ,fact_notes10           varchar (1000)    default 'unknown'
      ,user_col_01            varchar (1000)    default 'unknown'
      ,user_col_02            varchar (1000)    default 'unknown'
      ,user_col_03            varchar (1000)    default 'unknown'
      ,user_col_04            varchar (1000)    default 'unknown'
      ,user_col_05            varchar (1000)    default 'unknown'
      ,user_col_06            varchar (1000)    default 'unknown'
      ,user_col_07            varchar (1000)    default 'unknown'
      ,user_col_08            varchar (1000)    default 'unknown'
      ,user_col_09            varchar (1000)    default 'unknown'
      ,user_col_10            varchar (1000)    default 'unknown'
)
```

4.2.46.6. ctl_dw_biz_reqts_doc_06

The first table stores the Major Measures workbook.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_measure_number	This is the Measure Number that will be used to identify the Measure on reports and in conversations.
measure_name	This is the Measure Name that will be used to identify the Measure on reports and in conversations.
measure_sdesc	This is the Measure Short Description that will be used to describe the Measure on reports and in conversations.
measure_ldesc	This is the Measure Long Description that will be used to describe the Measure on reports and in conversations.
owner_name	This is the Measure Owner Name that will be used to document the Measure on reports.
owner_department	This is the Measure Owner Department that will be used to document the Measure on reports.
measure_importance	This is the Measure Importance that will be used to document the Measure Importance on reports.
measure_priority	This is the Measure Priority that will be used to document the Measure Priority on reports.
planned_iteration	This is the Measure Delivered In Iteration information that will be used to document the Measure Delivered In Iteration information on reports.
measure_status	This is the Measure Status information that will be used to document the Measure Status information on reports.
documentation_link	This is the Measure Link to Documentation information that will be used to document the Measure Link to Documentation information on reports.
last_review_date	This is the Measure Last Review Date information that will be used to document the Measure Last Review Date information on reports.
measure_notes01	This is the Measure Notes1 information that will be used to document the Measure Notes1 information on reports.
measure_notes02	This is the Measure Notes2 information that will be used to document the Measure Notes2 information on reports.
measure_notes03	This is the Measure Notes3 information that will be used to document the Measure Notes3 information on reports.
measure_notes04	This is the Measure Notes4 information that will be used to document the Measure Notes4 information on reports.
measure_notes05	This is the Measure Notes5 information that will be used to document the Measure Notes5 information on reports.
measure_notes06	This is the Measure Notes6 information that will be used to document the Measure Notes6 information on reports.
measure_notes07	This is the Measure Notes7 information that will be used to document the Measure Notes7 information on reports.
measure_notes08	This is the Measure Notes8 information that will be used to document the Measure Notes8 information on reports.
measure_notes09	This is the Measure Notes9 information that will be used to document the Measure Notes9 information on reports.
measure_notes10	This is the Measure Notes10 information that will be used to document the Measure Notes10 information on reports.
user_col_01	This is the Measure UserCol01 information that will be used to document the Measure UserCol01 information on reports.
user_col_02	This is the Measure UserCol02 information that will be used to document the Measure UserCol02 information on reports.
user_col_03	This is the Measure UserCol03 information that will be used to document the

	Measure UserCol03 information on reports.
user_col_04	This is the Measure UserCol04 information that will be used to document the Measure UserCol04 information on reports.
user_col_05	This is the Measure UserCol05 information that will be used to document the Measure UserCol05 information on reports.
user_col_06	This is the Measure UserCol06 information that will be used to document the Measure UserCol06 information on reports.
user_col_07	This is the Measure UserCol07 information that will be used to document the Measure UserCol07 information on reports.
user_col_08	This is the Measure UserCol08 information that will be used to document the Measure UserCol08 information on reports.
user_col_09	This is the Measure UserCol09 information that will be used to document the Measure UserCol09 information on reports.
user_col_10	This is the Measure UserCol10 information that will be used to document the Measure UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_06 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_measures_number     varchar (255)     not null default 'unknown'
      ,measures_name          varchar (255)     not null default 'unknown'
      ,measures_sdesc         varchar (255)     not null default 'unknown'
      ,measures_ldesc         varchar (2000)    not null default 'unknown'
      ,owner_name             varchar (255)     not null default 'unknown'
      ,owner_department       varchar (255)     not null default 'unknown'
      ,measures_importance    varchar (255)     not null default 'unknown'
      ,measures_priority      varchar (255)     not null default 'unknown'
      ,planned_iteration      varchar (255)     not null default 'unknown'
      ,measures_status        varchar (255)     not null default 'unknown'
      ,documentation_link     varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,measures_notes01       varchar (1000)    default 'unknown'
      ,measures_notes02       varchar (1000)    default 'unknown'
      ,measures_notes03       varchar (1000)    default 'unknown'
      ,measures_notes04       varchar (1000)    default 'unknown'
      ,measures_notes05       varchar (1000)    default 'unknown'
      ,measures_notes06       varchar (1000)    default 'unknown'
      ,measures_notes07       varchar (1000)    default 'unknown'
      ,measures_notes08       varchar (1000)    default 'unknown'
      ,measures_notes09       varchar (1000)    default 'unknown'
      ,measures_notes10       varchar (1000)    default 'unknown'
      ,user_col_01            varchar (1000)    default 'unknown'
      ,user_col_02            varchar (1000)    default 'unknown'
      ,user_col_03            varchar (1000)    default 'unknown'
      ,user_col_04            varchar (1000)    default 'unknown'
      ,user_col_05            varchar (1000)    default 'unknown'
      ,user_col_06            varchar (1000)    default 'unknown'
      ,user_col_07            varchar (1000)    default 'unknown'
      ,user_col_08            varchar (1000)    default 'unknown'
      ,user_col_09            varchar (1000)    default 'unknown'
      ,user_col_10            varchar (1000)    default 'unknown'
)
```

4.2.46.7. ctl_dw_biz_reqts_doc_07

The first table stores the Subject Area workbook.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_subject_area_number	This is the Subject Area Number that will be used to identify the Subject Area on reports and in conversations.
subject_area_name	This is the Subject Area Name that will be used to identify the Subject Area on reports and in conversations.
subject_area_sdesc	This is the Subject Area Short Description that will be used to describe the Subject Area on reports and in conversations.
subject_area_ldesc	This is the Subject Area Long Description that will be used to describe the Subject Area on reports and in conversations.
owner_name	This is the Subject Area Owner Name that will be used to document the Subject Area on reports.
owner_department	This is the Subject Area Owner Department that will be used to document the Subject Area on reports.
subject_area_importance	This is the Subject Area Importance that will be used to document the Subject Area Importance on reports.
subject_area_priority	This is the Subject Area Priority that will be used to document the Subject Area Priority on reports.
planned_iteration	This is the Subject Area Delivered In Iteration information that will be used to document the Subject Area Delivered In Iteration information on reports.
subject_area_status	This is the Subject Area Status information that will be used to document the Subject Area Status information on reports.
documentation_link	This is the Subject Area Link to Documentation information that will be used to document the Subject Area Link to Documentation information on reports.
last_review_date	This is the Subject Area Last Review Date information that will be used to document the Subject Area Last Review Date information on reports.
subject_area_notes01	This is the Subject Area Notes1 information that will be used to document the Subject Area Notes1 information on reports.
subject_area_notes02	This is the Subject Area Notes2 information that will be used to document the Subject Area Notes2 information on reports.
subject_area_notes03	This is the Subject Area Notes3 information that will be used to document the Subject Area Notes3 information on reports.
subject_area_notes04	This is the Subject Area Notes4 information that will be used to document the Subject Area Notes4 information on reports.
subject_area_notes05	This is the Subject Area Notes5 information that will be used to document the Subject Area Notes5 information on reports.
subject_area_notes06	This is the Subject Area Notes6 information that will be used to document the Subject Area Notes6 information on reports.
subject_area_notes07	This is the Subject Area Notes7 information that will be used to document the Subject Area Notes7 information on reports.
subject_area_notes08	This is the Subject Area Notes8 information that will be used to document the Subject Area Notes8 information on reports.
subject_area_notes09	This is the Subject Area Notes9 information that will be used to document the Subject Area Notes9 information on reports.
subject_area_notes10	This is the Subject Area Notes10 information that will be used to document the Subject Area Notes10 information on reports.
user_col_01	This is the Subject Area UserCol01 information that will be used to document the Subject Area UserCol01 information on reports.
user_col_02	This is the Subject Area UserCol02 information that will be used to document the Subject Area UserCol02 information on reports.
user_col_03	This is the Subject Area UserCol03 information that will be used to document the

	Subject Area UserCol03 information on reports.
user_col_04	This is the Subject Area UserCol04 information that will be used to document the Subject Area UserCol04 information on reports.
user_col_05	This is the Subject Area UserCol05 information that will be used to document the Subject Area UserCol05 information on reports.
user_col_06	This is the Subject Area UserCol06 information that will be used to document the Subject Area UserCol06 information on reports.
user_col_07	This is the Subject Area UserCol07 information that will be used to document the Subject Area UserCol07 information on reports.
user_col_08	This is the Subject Area UserCol08 information that will be used to document the Subject Area UserCol08 information on reports.
user_col_09	This is the Subject Area UserCol09 information that will be used to document the Subject Area UserCol09 information on reports.
user_col_10	This is the Subject Area UserCol10 information that will be used to document the Subject Area UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_07 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_subject_area_number varchar (255)     not null default 'unknown'
      ,subject_area_name      varchar (255)     not null default 'unknown'
      ,subject_area_sdesc     varchar (255)     not null default 'unknown'
      ,subject_area_ldesc     varchar (2000)    not null default 'unknown'
      ,owner_name             varchar (255)     not null default 'unknown'
      ,owner_department       varchar (255)     not null default 'unknown'
      ,subject_area_importance varchar (255)    not null default 'unknown'
      ,subject_area_priority  varchar (255)     not null default 'unknown'
      ,planned_iteration      varchar (255)     not null default 'unknown'
      ,subject_area_status    varchar (255)     not null default 'unknown'
      ,documentation_link     varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,subject_area_notes01   varchar (1000)    default 'unknown'
      ,subject_area_notes02   varchar (1000)    default 'unknown'
      ,subject_area_notes03   varchar (1000)    default 'unknown'
      ,subject_area_notes04   varchar (1000)    default 'unknown'
      ,subject_area_notes05   varchar (1000)    default 'unknown'
      ,subject_area_notes06   varchar (1000)    default 'unknown'
      ,subject_area_notes07   varchar (1000)    default 'unknown'
      ,subject_area_notes08   varchar (1000)    default 'unknown'
      ,subject_area_notes09   varchar (1000)    default 'unknown'
      ,subject_area_notes10   varchar (1000)    default 'unknown'
      ,user_col_01            varchar (1000)    default 'unknown'
      ,user_col_02            varchar (1000)    default 'unknown'
      ,user_col_03            varchar (1000)    default 'unknown'
      ,user_col_04            varchar (1000)    default 'unknown'
      ,user_col_05            varchar (1000)    default 'unknown'
      ,user_col_06            varchar (1000)    default 'unknown'
      ,user_col_07            varchar (1000)    default 'unknown'
      ,user_col_08            varchar (1000)    default 'unknown'
      ,user_col_09            varchar (1000)    default 'unknown'
      ,user_col_10            varchar (1000)    default 'unknown'
)
```

4.2.46.8. ctl_dw_biz_reqts_doc_51

The first table stores the Cross Reference 01 workbook which cross references project objectives and business requirements.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_xref_number	This is the Cross Reference Number that will be used to identify the Cross Reference on reports and in conversations.
project_objective_number	This is the Project Objective Number that will be used to identify the Project Objective on reports and in conversations.
project_objective_name	This is the Project Objective Name that will be used to identify the Project Objective on reports and in conversations.
biz_reqt_number	This is the Business Requirement Number that will be used to identify the Business Requirement on reports and in conversations.
biz_reqt_name	This is the Business Requirement Name that will be used to identify the Business Requirement on reports and in conversations.
xref_importance	This is the Cross Reference Importance that will be used to document the Cross Reference Importance on reports.
xref_priority	This is the Cross Reference Priority that will be used to document the Cross Reference Priority on reports.
xref_status	This is the Cross Reference Status information that will be used to document the Cross Reference Status information on reports.
documentation_link	This is the Cross Reference Link to Documentation information that will be used to document the Cross Reference Link to Documentation information on reports.
last_review_date	This is the Cross Reference Last Review Date information that will be used to document the Cross Reference Last Review Date information on reports.
xref_notes01	This is the Cross Reference Notes1 information that will be used to document the Cross Reference Notes1 information on reports.
xref_notes02	This is the Cross Reference Notes2 information that will be used to document the Cross Reference Notes2 information on reports.
xref_notes03	This is the Cross Reference Notes3 information that will be used to document the Cross Reference Notes3 information on reports.
xref_notes04	This is the Cross Reference Notes4 information that will be used to document the Cross Reference Notes4 information on reports.
xref_notes05	This is the Cross Reference Notes5 information that will be used to document the Cross Reference Notes5 information on reports.
xref_notes06	This is the Cross Reference Notes6 information that will be used to document the Cross Reference Notes6 information on reports.
xref_notes07	This is the Cross Reference Notes7 information that will be used to document the Cross Reference Notes7 information on reports.
xref_notes08	This is the Cross Reference Notes8 information that will be used to document the Cross Reference Notes8 information on reports.
xref_notes09	This is the Cross Reference Notes9 information that will be used to document the Cross Reference Notes9 information on reports.
xref_notes10	This is the Cross Reference Notes10 information that will be used to document the Cross Reference Notes10 information on reports.
user_col_01	This is the Cross Reference UserCol01 information that will be used to document the Cross Reference UserCol01 information on reports.
user_col_02	This is the Cross Reference UserCol02 information that will be used to document the Cross Reference UserCol02 information on reports.
user_col_03	This is the Cross Reference UserCol03 information that will be used to document the Cross Reference UserCol03 information on reports.
user_col_04	This is the Cross Reference UserCol04 information that will be used to document the Cross Reference UserCol04 information on reports.

user_col_05	This is the Cross Reference UserCol05 information that will be used to document the Cross Reference UserCol05 information on reports.
user_col_06	This is the Cross Reference UserCol06 information that will be used to document the Cross Reference UserCol06 information on reports.
user_col_07	This is the Cross Reference UserCol07 information that will be used to document the Cross Reference UserCol07 information on reports.
user_col_08	This is the Cross Reference UserCol08 information that will be used to document the Cross Reference UserCol08 information on reports.
user_col_09	This is the Cross Reference UserCol09 information that will be used to document the Cross Reference UserCol09 information on reports.
user_col_10	This is the Cross Reference UserCol10 information that will be used to document the Cross Reference UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_51 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_xref_number         varchar (255)     not null default 'unknown'
      ,project_objective_number varchar (255)   not null default 'unknown'
      ,project_objective_name varchar (255)     not null default 'unknown'
      ,biz_reqt_number        varchar (255)     not null default 'unknown'
      ,biz_reqt_name          varchar (255)     not null default 'unknown'
      ,xref_importance        varchar (255)     not null default 'unknown'
      ,xref_priority          varchar (255)     not null default 'unknown'
      ,xref_status            varchar (255)     not null default 'unknown'
      ,documentation_link     varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,xref_notes01           varchar (1000)    default 'unknown'
      ,xref_notes02           varchar (1000)    default 'unknown'
      ,xref_notes03           varchar (1000)    default 'unknown'
      ,xref_notes04           varchar (1000)    default 'unknown'
      ,xref_notes05           varchar (1000)    default 'unknown'
      ,xref_notes06           varchar (1000)    default 'unknown'
      ,xref_notes07           varchar (1000)    default 'unknown'
      ,xref_notes08           varchar (1000)    default 'unknown'
      ,xref_notes09           varchar (1000)    default 'unknown'
      ,xref_notes10           varchar (1000)    default 'unknown'
      ,user_col_01            varchar (1000)    default 'unknown'
      ,user_col_02            varchar (1000)    default 'unknown'
      ,user_col_03            varchar (1000)    default 'unknown'
      ,user_col_04            varchar (1000)    default 'unknown'
      ,user_col_05            varchar (1000)    default 'unknown'
      ,user_col_06            varchar (1000)    default 'unknown'
      ,user_col_07            varchar (1000)    default 'unknown'
      ,user_col_08            varchar (1000)    default 'unknown'
      ,user_col_09            varchar (1000)    default 'unknown'
      ,user_col_10            varchar (1000)    default 'unknown'
)
```

4.2.46.9. ctl_dw_biz_reqts_doc_52

The first table stores the Cross Reference 02 workbook which cross references project objectives, business requirements and dimensions.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_xref_number	This is the Cross Reference Number that will be used to identify the Cross Reference on reports and in conversations.
project_objective_number	This is the Project Objective Number that will be used to identify the Project Objective on reports and in conversations.
project_objective_name	This is the Project Objective Name that will be used to identify the Project Objective on reports and in conversations.
biz_reqt_number	This is the Business Requirement Number that will be used to identify the Business Requirement on reports and in conversations.
biz_reqt_name	This is the Business Requirement Name that will be used to identify the Business Requirement on reports and in conversations.
dimension_number	This is the Dimension Number that will be used to identify the Business Requirement on reports and in conversations.
dimension_name	This is the Dimension Name that will be used to identify the Business Requirement on reports and in conversations.
xref_importance	This is the Cross Reference Importance that will be used to document the Cross Reference Importance on reports.
xref_priority	This is the Cross Reference Priority that will be used to document the Cross Reference Priority on reports.
xref_status	This is the Cross Reference Status information that will be used to document the Cross Reference Status information on reports.
documentation_link	This is the Cross Reference Link to Documentation information that will be used to document the Cross Reference Link to Documentation information on reports.
last_review_date	This is the Cross Reference Last Review Date information that will be used to document the Cross Reference Last Review Date information on reports.
xref_notes01	This is the Cross Reference Notes1 information that will be used to document the Cross Reference Notes1 information on reports.
xref_notes02	This is the Cross Reference Notes2 information that will be used to document the Cross Reference Notes2 information on reports.
xref_notes03	This is the Cross Reference Notes3 information that will be used to document the Cross Reference Notes3 information on reports.
xref_notes04	This is the Cross Reference Notes4 information that will be used to document the Cross Reference Notes4 information on reports.
xref_notes05	This is the Cross Reference Notes5 information that will be used to document the Cross Reference Notes5 information on reports.
xref_notes06	This is the Cross Reference Notes6 information that will be used to document the Cross Reference Notes6 information on reports.
xref_notes07	This is the Cross Reference Notes7 information that will be used to document the Cross Reference Notes7 information on reports.
xref_notes08	This is the Cross Reference Notes8 information that will be used to document the Cross Reference Notes8 information on reports.
xref_notes09	This is the Cross Reference Notes9 information that will be used to document the Cross Reference Notes9 information on reports.
xref_notes10	This is the Cross Reference Notes10 information that will be used to document the Cross Reference Notes10 information on reports.
user_col_01	This is the Cross Reference UserCol01 information that will be used to document the Cross Reference UserCol01 information on reports.
user_col_02	This is the Cross Reference UserCol02 information that will be used to document the Cross Reference UserCol02 information on reports.

user_col_03	This is the Cross Reference UserCol03 information that will be used to document the Cross Reference UserCol03 information on reports.
user_col_04	This is the Cross Reference UserCol04 information that will be used to document the Cross Reference UserCol04 information on reports.
user_col_05	This is the Cross Reference UserCol05 information that will be used to document the Cross Reference UserCol05 information on reports.
user_col_06	This is the Cross Reference UserCol06 information that will be used to document the Cross Reference UserCol06 information on reports.
user_col_07	This is the Cross Reference UserCol07 information that will be used to document the Cross Reference UserCol07 information on reports.
user_col_08	This is the Cross Reference UserCol08 information that will be used to document the Cross Reference UserCol08 information on reports.
user_col_09	This is the Cross Reference UserCol09 information that will be used to document the Cross Reference UserCol09 information on reports.
user_col_10	This is the Cross Reference UserCol10 information that will be used to document the Cross Reference UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_52 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_xref_number         varchar (255)     not null default 'unknown'
      ,project_objective_number varchar (255)   not null default 'unknown'
      ,project_objective_name varchar (255)     not null default 'unknown'
      ,biz_reqt_number        varchar (255)     not null default 'unknown'
      ,biz_reqt_name          varchar (255)     not null default 'unknown'
      ,dimension_number       varchar (255)     not null default 'unknown'
      ,dimension_name         varchar (255)     not null default 'unknown'
      ,xref_importance        varchar (255)     not null default 'unknown'
      ,xref_priority          varchar (255)     not null default 'unknown'
      ,xref_status            varchar (255)     not null default 'unknown'
      ,documentation_link     varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,xref_notes01           varchar (1000)    default 'unknown'
      ,xref_notes02           varchar (1000)    default 'unknown'
      ,xref_notes03           varchar (1000)    default 'unknown'
      ,xref_notes04           varchar (1000)    default 'unknown'
      ,xref_notes05           varchar (1000)    default 'unknown'
      ,xref_notes06           varchar (1000)    default 'unknown'
      ,xref_notes07           varchar (1000)    default 'unknown'
      ,xref_notes08           varchar (1000)    default 'unknown'
      ,xref_notes09           varchar (1000)    default 'unknown'
      ,xref_notes10           varchar (1000)    default 'unknown'
      ,user_col_01            varchar (1000)    default 'unknown'
      ,user_col_02            varchar (1000)    default 'unknown'
      ,user_col_03            varchar (1000)    default 'unknown'
      ,user_col_04            varchar (1000)    default 'unknown'
      ,user_col_05            varchar (1000)    default 'unknown'
      ,user_col_06            varchar (1000)    default 'unknown'
      ,user_col_07            varchar (1000)    default 'unknown'
      ,user_col_08            varchar (1000)    default 'unknown'
      ,user_col_09            varchar (1000)    default 'unknown'
      ,user_col_10            varchar (1000)    default 'unknown'
)
```

4.2.46.10.ctl_dw_biz_reqts_doc_53

The first table stores the Cross Reference 03 workbook which cross references project objectives, business requirements and major facts.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_xref_number	This is the Cross Reference Number that will be used to identify the Cross Reference on reports and in conversations.
project_objective_number	This is the Project Objective Number that will be used to identify the Project Objective on reports and in conversations.
project_objective_name	This is the Project Objective Name that will be used to identify the Project Objective on reports and in conversations.
biz_reqt_number	This is the Business Requirement Number that will be used to identify the Business Requirement on reports and in conversations.
biz_reqt_name	This is the Business Requirement Name that will be used to identify the Business Requirement on reports and in conversations.
fact_number	This is the Fact Number that will be used to identify the Business Requirement on reports and in conversations.
fact_name	This is the Fact Name that will be used to identify the Business Requirement on reports and in conversations.
xref_importance	This is the Cross Reference Importance that will be used to document the Cross Reference Importance on reports.
xref_priority	This is the Cross Reference Priority that will be used to document the Cross Reference Priority on reports.
xref_status	This is the Cross Reference Status information that will be used to document the Cross Reference Status information on reports.
documentation_link	This is the Cross Reference Link to Documentation information that will be used to document the Cross Reference Link to Documentation information on reports.
last_review_date	This is the Cross Reference Last Review Date information that will be used to document the Cross Reference Last Review Date information on reports.
xref_notes01	This is the Cross Reference Notes1 information that will be used to document the Cross Reference Notes1 information on reports.
xref_notes02	This is the Cross Reference Notes2 information that will be used to document the Cross Reference Notes2 information on reports.
xref_notes03	This is the Cross Reference Notes3 information that will be used to document the Cross Reference Notes3 information on reports.
xref_notes04	This is the Cross Reference Notes4 information that will be used to document the Cross Reference Notes4 information on reports.
xref_notes05	This is the Cross Reference Notes5 information that will be used to document the Cross Reference Notes5 information on reports.
xref_notes06	This is the Cross Reference Notes6 information that will be used to document the Cross Reference Notes6 information on reports.
xref_notes07	This is the Cross Reference Notes7 information that will be used to document the Cross Reference Notes7 information on reports.
xref_notes08	This is the Cross Reference Notes8 information that will be used to document the Cross Reference Notes8 information on reports.
xref_notes09	This is the Cross Reference Notes9 information that will be used to document the Cross Reference Notes9 information on reports.
xref_notes10	This is the Cross Reference Notes10 information that will be used to document the Cross Reference Notes10 information on reports.
user_col_01	This is the Cross Reference UserCol01 information that will be used to document the Cross Reference UserCol01 information on reports.
user_col_02	This is the Cross Reference UserCol02 information that will be used to document the Cross Reference UserCol02 information on reports.

user_col_03	This is the Cross Reference UserCol03 information that will be used to document the Cross Reference UserCol03 information on reports.
user_col_04	This is the Cross Reference UserCol04 information that will be used to document the Cross Reference UserCol04 information on reports.
user_col_05	This is the Cross Reference UserCol05 information that will be used to document the Cross Reference UserCol05 information on reports.
user_col_06	This is the Cross Reference UserCol06 information that will be used to document the Cross Reference UserCol06 information on reports.
user_col_07	This is the Cross Reference UserCol07 information that will be used to document the Cross Reference UserCol07 information on reports.
user_col_08	This is the Cross Reference UserCol08 information that will be used to document the Cross Reference UserCol08 information on reports.
user_col_09	This is the Cross Reference UserCol09 information that will be used to document the Cross Reference UserCol09 information on reports.
user_col_10	This is the Cross Reference UserCol10 information that will be used to document the Cross Reference UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_53 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_xref_number         varchar (255)     not null default 'unknown'
      ,project_objective_number varchar (255)   not null default 'unknown'
      ,project_objective_name varchar (255)     not null default 'unknown'
      ,biz_reqt_number        varchar (255)     not null default 'unknown'
      ,biz_reqt_name          varchar (255)     not null default 'unknown'
      ,fact_number            varchar (255)     not null default 'unknown'
      ,fact_name              varchar (255)     not null default 'unknown'
      ,xref_importance        varchar (255)     not null default 'unknown'
      ,xref_priority          varchar (255)     not null default 'unknown'
      ,xref_status            varchar (255)     not null default 'unknown'
      ,documentation_link     varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,xref_notes01           varchar (1000)    default 'unknown'
      ,xref_notes02           varchar (1000)    default 'unknown'
      ,xref_notes03           varchar (1000)    default 'unknown'
      ,xref_notes04           varchar (1000)    default 'unknown'
      ,xref_notes05           varchar (1000)    default 'unknown'
      ,xref_notes06           varchar (1000)    default 'unknown'
      ,xref_notes07           varchar (1000)    default 'unknown'
      ,xref_notes08           varchar (1000)    default 'unknown'
      ,xref_notes09           varchar (1000)    default 'unknown'
      ,xref_notes10           varchar (1000)    default 'unknown'
      ,user_col_01            varchar (1000)    default 'unknown'
      ,user_col_02            varchar (1000)    default 'unknown'
      ,user_col_03            varchar (1000)    default 'unknown'
      ,user_col_04            varchar (1000)    default 'unknown'
      ,user_col_05            varchar (1000)    default 'unknown'
      ,user_col_06            varchar (1000)    default 'unknown'
      ,user_col_07            varchar (1000)    default 'unknown'
      ,user_col_08            varchar (1000)    default 'unknown'
      ,user_col_09            varchar (1000)    default 'unknown'
      ,user_col_10            varchar (1000)    default 'unknown'
)
```

4.2.46.11.ctl_dw_biz_reqts_doc_54

The first table stores the Cross Reference 04 workbook which cross references project objective, business requirements and measures.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_xref_number	This is the Cross Reference Number that will be used to identify the Cross Reference on reports and in conversations.
project_objective_number	This is the Project Objective Number that will be used to identify the Project Objective on reports and in conversations.
project_objective_name	This is the Project Objective Name that will be used to identify the Project Objective on reports and in conversations.
biz_reqt_number	This is the Business Requirement Number that will be used to identify the Business Requirement on reports and in conversations.
biz_reqt_name	This is the Business Requirement Name that will be used to identify the Business Requirement on reports and in conversations.
Measure_number	This is the Measure Number that will be used to identify the Business Requirement on reports and in conversations.
measure_name	This is the Measure Name that will be used to identify the Business Requirement on reports and in conversations.
xref_importance	This is the Cross Reference Importance that will be used to document the Cross Reference Importance on reports.
xref_priority	This is the Cross Reference Priority that will be used to document the Cross Reference Priority on reports.
xref_status	This is the Cross Reference Status information that will be used to document the Cross Reference Status information on reports.
documentation_link	This is the Cross Reference Link to Documentation information that will be used to document the Cross Reference Link to Documentation information on reports.
last_review_date	This is the Cross Reference Last Review Date information that will be used to document the Cross Reference Last Review Date information on reports.
xref_notes01	This is the Cross Reference Notes1 information that will be used to document the Cross Reference Notes1 information on reports.
xref_notes02	This is the Cross Reference Notes2 information that will be used to document the Cross Reference Notes2 information on reports.
xref_notes03	This is the Cross Reference Notes3 information that will be used to document the Cross Reference Notes3 information on reports.
xref_notes04	This is the Cross Reference Notes4 information that will be used to document the Cross Reference Notes4 information on reports.
xref_notes05	This is the Cross Reference Notes5 information that will be used to document the Cross Reference Notes5 information on reports.
xref_notes06	This is the Cross Reference Notes6 information that will be used to document the Cross Reference Notes6 information on reports.
xref_notes07	This is the Cross Reference Notes7 information that will be used to document the Cross Reference Notes7 information on reports.
xref_notes08	This is the Cross Reference Notes8 information that will be used to document the Cross Reference Notes8 information on reports.
xref_notes09	This is the Cross Reference Notes9 information that will be used to document the Cross Reference Notes9 information on reports.
xref_notes10	This is the Cross Reference Notes10 information that will be used to document the Cross Reference Notes10 information on reports.
user_col_01	This is the Cross Reference UserCol01 information that will be used to document the Cross Reference UserCol01 information on reports.
user_col_02	This is the Cross Reference UserCol02 information that will be used to document the Cross Reference UserCol02 information on reports.

user_col_03	This is the Cross Reference UserCol03 information that will be used to document the Cross Reference UserCol03 information on reports.
user_col_04	This is the Cross Reference UserCol04 information that will be used to document the Cross Reference UserCol04 information on reports.
user_col_05	This is the Cross Reference UserCol05 information that will be used to document the Cross Reference UserCol05 information on reports.
user_col_06	This is the Cross Reference UserCol06 information that will be used to document the Cross Reference UserCol06 information on reports.
user_col_07	This is the Cross Reference UserCol07 information that will be used to document the Cross Reference UserCol07 information on reports.
user_col_08	This is the Cross Reference UserCol08 information that will be used to document the Cross Reference UserCol08 information on reports.
user_col_09	This is the Cross Reference UserCol09 information that will be used to document the Cross Reference UserCol09 information on reports.
user_col_10	This is the Cross Reference UserCol10 information that will be used to document the Cross Reference UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_53 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_xref_number         varchar (255)     not null default 'unknown'
      ,project_objective_number varchar (255)   not null default 'unknown'
      ,project_objective_name varchar (255)     not null default 'unknown'
      ,biz_reqt_number        varchar (255)     not null default 'unknown'
      ,biz_reqt_name          varchar (255)     not null default 'unknown'
      ,measure_number         varchar (255)     not null default 'unknown'
      ,measure_name           varchar (255)     not null default 'unknown'
      ,xref_importance        varchar (255)     not null default 'unknown'
      ,xref_priority          varchar (255)     not null default 'unknown'
      ,xref_status            varchar (255)     not null default 'unknown'
      ,documentation_link     varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,xref_notes01           varchar (1000)    default 'unknown'
      ,xref_notes02           varchar (1000)    default 'unknown'
      ,xref_notes03           varchar (1000)    default 'unknown'
      ,xref_notes04           varchar (1000)    default 'unknown'
      ,xref_notes05           varchar (1000)    default 'unknown'
      ,xref_notes06           varchar (1000)    default 'unknown'
      ,xref_notes07           varchar (1000)    default 'unknown'
      ,xref_notes08           varchar (1000)    default 'unknown'
      ,xref_notes09           varchar (1000)    default 'unknown'
      ,xref_notes10           varchar (1000)    default 'unknown'
      ,user_col_01            varchar (1000)    default 'unknown'
      ,user_col_02            varchar (1000)    default 'unknown'
      ,user_col_03            varchar (1000)    default 'unknown'
      ,user_col_04            varchar (1000)    default 'unknown'
      ,user_col_05            varchar (1000)    default 'unknown'
      ,user_col_06            varchar (1000)    default 'unknown'
      ,user_col_07            varchar (1000)    default 'unknown'
      ,user_col_08            varchar (1000)    default 'unknown'
      ,user_col_09            varchar (1000)    default 'unknown'
      ,user_col_10            varchar (1000)    default 'unknown'
)
```

4.2.46.12.ctl_dw_biz_reqts_doc_55

The first table stores the Cross Reference 05 workbook which cross references subject areas to tables.

Column Name	Function
pk_spreadsheet_name	The name of the spreadsheet from which this table row was loaded.
pk_spreadsheet_version	The version of the spreadsheet from which this table row was loaded.
pk_row_number	This is a row number that is placed onto the row to make sure that the row is unique.
pk_xref_number	This is the Cross Reference Number that will be used to identify the Cross Reference on reports and in conversations.
subject_area_number	This is the Subject Area Number that will be used to identify the Subject Area on reports and in conversations.
subject_area_name	This is the Subject Area Name that will be used to identify the Subject Area on reports and in conversations.
table_number	This is the Table Number that will be used to identify the Table on reports and in conversations.
table_name	This is the Table Name that will be used to identify the Table on reports and in conversations.
xref_importance	This is the Cross Reference Importance that will be used to document the Cross Reference Importance on reports.
xref_priority	This is the Cross Reference Priority that will be used to document the Cross Reference Priority on reports.
xref_status	This is the Cross Reference Status information that will be used to document the Cross Reference Status information on reports.
documentation_link	This is the Cross Reference Link to Documentation information that will be used to document the Cross Reference Link to Documentation information on reports.
last_review_date	This is the Cross Reference Last Review Date information that will be used to document the Cross Reference Last Review Date information on reports.
xref_notes01	This is the Cross Reference Notes1 information that will be used to document the Cross Reference Notes1 information on reports.
xref_notes02	This is the Cross Reference Notes2 information that will be used to document the Cross Reference Notes2 information on reports.
xref_notes03	This is the Cross Reference Notes3 information that will be used to document the Cross Reference Notes3 information on reports.
xref_notes04	This is the Cross Reference Notes4 information that will be used to document the Cross Reference Notes4 information on reports.
xref_notes05	This is the Cross Reference Notes5 information that will be used to document the Cross Reference Notes5 information on reports.
xref_notes06	This is the Cross Reference Notes6 information that will be used to document the Cross Reference Notes6 information on reports.
xref_notes07	This is the Cross Reference Notes7 information that will be used to document the Cross Reference Notes7 information on reports.
xref_notes08	This is the Cross Reference Notes8 information that will be used to document the Cross Reference Notes8 information on reports.
xref_notes09	This is the Cross Reference Notes9 information that will be used to document the Cross Reference Notes9 information on reports.
xref_notes10	This is the Cross Reference Notes10 information that will be used to document the Cross Reference Notes10 information on reports.
user_col_01	This is the Cross Reference UserCol01 information that will be used to document the Cross Reference UserCol01 information on reports.
user_col_02	This is the Cross Reference UserCol02 information that will be used to document the Cross Reference UserCol02 information on reports.
user_col_03	This is the Cross Reference UserCol03 information that will be used to document the Cross Reference UserCol03 information on reports.
user_col_04	This is the Cross Reference UserCol04 information that will be used to document the Cross Reference UserCol04 information on reports.
user_col_05	This is the Cross Reference UserCol05 information that will be used to document

	the Cross Reference UserCol05 information on reports.
user_col_06	This is the Cross Reference UserCol06 information that will be used to document the Cross Reference UserCol06 information on reports.
user_col_07	This is the Cross Reference UserCol07 information that will be used to document the Cross Reference UserCol07 information on reports.
user_col_08	This is the Cross Reference UserCol08 information that will be used to document the Cross Reference UserCol08 information on reports.
user_col_09	This is the Cross Reference UserCol09 information that will be used to document the Cross Reference UserCol09 information on reports.
user_col_10	This is the Cross Reference UserCol10 information that will be used to document the Cross Reference UserCol10 information on reports.

Example DDL for the table is as follows:

```
create table dbo.ctl_dw_biz_reqts_doc_55 (
       pk_spreadsheet_name    varchar (255)     default 'unknown'
      ,pk_spreadsheet_version varchar (255)     default 'unknown'
      ,pk_row_number          integer           not null default 0
      ,pk_xref_number         varchar (255)     not null default 'unknown'
      ,subject_area_number    varchar (255)     not null default 'unknown'
      ,subject_area_name      varchar (255)     not null default 'unknown'
      ,table_number           varchar (255)     not null default 'unknown'
      ,table_name             varchar (255)     not null default 'unknown'
      ,xref_importance        varchar (255)     not null default 'unknown'
      ,xref_priority          varchar (255)     not null default 'unknown'
      ,xref_status            varchar (255)     not null default 'unknown'
      ,documentation_link     varchar (255)     not null default 'unknown'
      ,last_review_date       datetime          not null default current_timestamp
      ,xref_notes01           varchar (1000)    default 'unknown'
      ,xref_notes02           varchar (1000)    default 'unknown'
      ,xref_notes03           varchar (1000)    default 'unknown'
      ,xref_notes04           varchar (1000)    default 'unknown'
      ,xref_notes05           varchar (1000)    default 'unknown'
      ,xref_notes06           varchar (1000)    default 'unknown'
      ,xref_notes07           varchar (1000)    default 'unknown'
      ,xref_notes08           varchar (1000)    default 'unknown'
      ,xref_notes09           varchar (1000)    default 'unknown'
      ,xref_notes10           varchar (1000)    default 'unknown'
      ,user_col_01            varchar (1000)    default 'unknown'
      ,user_col_02            varchar (1000)    default 'unknown'
      ,user_col_03            varchar (1000)    default 'unknown'
      ,user_col_04            varchar (1000)    default 'unknown'
      ,user_col_05            varchar (1000)    default 'unknown'
      ,user_col_06            varchar (1000)    default 'unknown'
      ,user_col_07            varchar (1000)    default 'unknown'
      ,user_col_08            varchar (1000)    default 'unknown'
      ,user_col_09            varchar (1000)    default 'unknown'
      ,user_col_10            varchar (1000)    default 'unknown'
)
```

4.2.46.13.ctl_proc_grp_start_run_log

This table stores log rows for each start of each process group run log. It was added to increase the details of the logging for process groups.

Column Name	Function
pk_batch_number	This is the number of the batch. It is a primary key for this table.
pk_process_batch_name	This is the batch name assigned for the specific batch. It is the same batch name as occurs on the ctl_batch_pre_req table. A Process Group is placed into a batch by using the same name that defines the batch. Example: batch001. Default: 'unknown'
pk_process_group_name	This is the name of the Process Group. The name of the process group links this Process to the Process Group. Example: group001, group002, group003 etc.
pk_started_tstamp	This is the start timestamp of the process group.
parent_process_id	This is the process id of the process group.
child_process_id	This is the process id of the child process that the process group spawns.
DBConnectionOutParameter	This is the ODBC connection that was in use to spawn the child process.
OutCatalogName	This is the out catalog name used to spawn the child process.
OutSchemaName	This is the out schema name used to spawn the child process.
OutTableName	This is the out table name used to spawn the child process.
TargetDataBase	This is the target Database Type that the child process will be spawned against.

Example DDL for the table is as follows:

```
create table dbo.ctl_proc_grp_start_run_log
(      pk_batch_number        integer                 not null    default 0
      ,pk_process_batch_name  varchar     (255)       not null    default 'unknown'
      ,pk_process_group_name  varchar     (255)       not null    default 'unknown'
      ,pk_started_tstamp      datetime
      ,parent_process_id      integer           not null    default 0
      ,child_process_id       integer           not null    default 0
      ,DBConnectionOutParameter varchar   (255)       not null    default 'unknown'
      ,OutCatalogName         varchar     (255)       not null    default 'unknown'
      ,OutSchemaName          varchar     (255)       not null    default 'unknown'
      ,OutTableName           varchar     (255)       not null    default 'unknown'
      ,TargetDataBase         varchar     (255)       not null    default 'unknown'
)
```

4.2.46.14.ctl_dw_delta_definitions

This table stores the definitions of the workbook that was used to create delta processing SQL. It is used for searching to save time looking up sql files etc. It is handy documentation for what was done in the delta definition workbook.

4.2.46.15.ctl_dw_nav_max_entry_nos

This table stores the maximum entry numbers from the ERP by MicroSoft called NAVision. It was developed for a client who used NAVision as their ERP to support the ETL process.

4.2.46.16.ctl_dw_nav_max_timestamps

This table stores the maximum timestamps from the ERP by MicroSoft called NAVision. It was developed for a client who used NAVision as their ERP to support the ETL process.

4.2.46.17.ctl_dw_nav_min_entry_nos

This table stores the minimum entry numbers from the ERP by MicroSoft called NAVision. It was developed for a client who used NAVision as their ERP to support the ETL process.

4.2.46.18.ctl_dw_nav_min_timestamps

This table stores the minimum timestamps from the ERP by MicroSoft called NAVision. It was developed for a client who used NAVision as their ERP to support the ETL process.

4.2.46.19.ctl_dw_pre_mappings_01

This table stores the Pre Mapping 01 information which is used to generate mappings to be placed in to the mapping spreadsheet.

4.2.46.20.ctl_src_table_defs

This table stores definitions of source tables to be sent in to the staging area.

4.2.46.21.ctl_src_column_defs

This table stores definitions of source columns to be sent in to the staging area.

4.2.46.22.ctl_src_index_defs

This table stores definitions of source system indexes to be used to determine primary keys for tables to be sent into the staging area.

4.2.46.23.ctl_src_to_stg_defs

This table stores high level maps of the source to staging area mappings. This is used to generate a number of workbooks such as the delta definition workbooks, and create tables workbooks.