The SeETL RunTime

User Guide

Appendices

Version 3.1.02
1st January 2014
Instant Business Intelligence
info@instantBI.com

Table of Contents

1. CHANGE CONTROL LOG

#	Date	Name	Description
1.0	12/1/03	IBI Dev	Initial version for publication with SeETL[RT].
1.1	12/1/03	IBI Dev	Documentation review and additional information based on testing of Oracle8 as a source and target for the data warehouse.
1.2	3/4/03	IBI Dev	Association tables were added to SeETL[RT]. The User Guide has been updated to reflect this new feature of SeETL[RT].
1.3	22/1/04	IBI Dev	Addition of Appendices explaining data warehouse data modelling in more detail.
1.4	25/3/04	IBI Dev	Consolidation of SeETL[RT] Utilities into the SeETL[RT] User Guide. Addition of: • Delimiter Separated Values Reformat Utility • Fixed File Format Reformat Utility • Load Interface File option of the Data Transfer Utility
1.5	2/4/04	IBI Dev	1. Addition of the documentation for the Generate Delta File Utility.
1.5.4	20/5/04	IBI Dev	1. Bring the version number of the documentation into line with the version number of the software. 2. Add the first released of the scheduler to the documentation. 3. Add the first release of the DataStage batch processing utility to the documentation. 4. Update the error messages table with the new error messages for the new utilities. 5. Allow the Data Transfer Utility to end with a zero return code even if there is no input data to the Data Transfer Utility.
1.5.5	30/6/04	IBI Dev	See Appendix 4 – Enhancements for Version 1.5.5
1.6.1	1/1/2005	IBI Dev	See Appendix 5 – Enhancements for Version 1.6.1 Appendices were split out from the full documentation in version 1.6.1. and they were also renumbered.
2.0	1/1/2005	IBI Dev	Migrated SeETL[RT] across to Instant Business Intelligence.
2.1	1/9/2005	IBI Dev	See Appendix 6 – Enhancements for Version 2.1
3.0	1/6/2006	IBI Dev	See Appendix 7 – Enhancements for Version 3.0
3.0.00	6/1/2008	IBI Dev	See Appendix 8 – Enhancements for Version 3.0.00
3.1.00	1/1/2012	IBI Dev	See Appendix 9 – Enhancements for Version 3.1.00
3.1.01	1/1/2013	IBI Dev	See Appendix 10 – Enhancements for Version 3.1.01
3.1.02	1/01/2014	IBI Dev	See Appendix 11 – Enhancements for Version 3.1.02

2. APPENDIX 1 - ERROR MESSAGES

This section documents the error messages that can be issued by SeETL[RT]. Note that in many cases you will see very similar if not exactly the same text printed as two different error messages. The purpose of this is to allow technical support to know exactly which piece of code issued the error message.

Message Number	Message Text
CTLI0001_MSG	Program starting.
CTLI0002_MSG	Program submitted with the following parameters
CTLI0003_MSG	Program ending.
CTLI0004_MSG	The following SQL Statement was successfully executed against the data warehouse database:
CTLI0005_MSG	The number of rows affected by the preceding SQL statement is as follows:
CTLI0006_MSG	An attempt has been made to re-start an abended batch. This message is provided for your information only. The select statement used to determine that the batch is in an abended state is as follows:"
CTLI0007_MSG	A DataStage job was successfully run. The details are as follows:"
CTLI0008_MSG	The number of rows transferred has exceeded the NumberRowsToTransfer parameter set by the user. Data Transfer will be stopped.
CTLI0009_MSG	The Dimension table requested to be loaded was already found loaded in a memory map. The select statement used to open the dimension table is as follows:
CTLI0010_MSG	The Dimension table was successfully loaded into a memory map. The size of the memory map and the sql used to load the memory map are as follows:
CTLI0011_MSG	The dimension tables that were specified to be loaded into memory maps have been loaded and CTLU012 has completed successfully. See error message log for details of the dimension tables loaded into memory maps.
CTLW0001_MSG	Error occurred when trying to execute a process group. The values of the batch number, batch name and process group are as follows: Batch Number=
CTLW0002_MSG	No pre-requisites found when performing the open for the dw proc grp pre req table. Though it is not an error to have a process group without pre-requisites it is quite unusual. The user should review the following select statement to make sure that the process group does not have pre-requisites. The select statement used to open the dw proc grp pre req table is as follows:
CTLW0003_MSG	A request has been made to restart a command in a process group that has already been completed in the current batch. The user should review the commands that have been completed in the process group to ensure that all processes have been completed successfully. The select statement used to detect this situation is as follows:
CTLW0004_MSG	A request has been made to start a batch as a one time batch that has already been started. If this batch is abended you must clear out all log records before you can re-run a one time batch. Details of the sql statement to detect that the one time batch has been previously run is as follows:
CTLW0005_MSG	The input table was empty and the InputTableOrFileCanBeEmpty parameter was set to Yes. The program was completed successfully. However the user should check to determine if an empty file condition is ok. The select statement which found no records in the input table is as follows:
CTLW0006_MSG	The input file CTLF001FileName was empty and the InputTableOrFileCanBeEmpty parameter was set to Yes. The program was completed successfully. However the user should check to determine if an empty file condition is ok. The CTLF001FileName is as follows:
CTLW0007_MSG	The input file CTLF002FileName was empty and the InputTableOrFileCanBeEmpty parameter was set to Yes. The program was completed successfully. However the user should check to determine if an empty file condition is ok. The CTLF002FileName is as follows:
CTLW0008_MSG	WorkFileName is empty and the InputTableOrFileCanBeEmpty parameter is set to Yes. The program was completed successfully. However the user should check to determine if an

	empty file condition is ok. The WorkFileName is as follows:
CTLW0009_MSG	The dimension table load control table contained no records indicating dimension tables should be loaded into memory mapped files and the InputTableOrFileCanBeEmpty parameter was set to Yes. The program was completed successfully. However the user should check to determine if an empty file condition is ok. The select statement which found no records in the dimension table load control table is as follows:
CTLW0010_MSG	A request has been made to restart a command in a process group that is still running in a current batch. The user should review the commands that have been completed in the process group to ensure that all processes have been completed successfully. The select statement used to detect this situation is as follows:
CTLW0011_MSG	A request has been made to restart a process group in a batch that is still running in a current batch. The user should review the process group run status to ensure this is correct. This message should only be issued when an attempt is made to restart a partially failed batch. This message should only be issued for process groups that continued running in the partially failed batch and therefore do not need to be restarted. The select statement used to detect this situation is as follows:
CTLW0012_MSG	An ODBC Data Cast Conversion Error has been detected and the AllowDataConversionError Parameter is set to Yes. Thus the data conversion error has been allowed and the record has been rejected. The row number of the record in the input file is as follows:
CTLW0013_MSG	A request has been made to start a process group in a batch that is waiting for pre-requisites. The user should review the process group run status to ensure this is correct. This message should only be issued when an attempt is made to restart a partially failed batch. This message should only be issued for process groups that were previously waiting in the partially failed batch and therefore do not need to be restarted. The select statement used to detect this situation is as follows:
CTLW0014_MSG	A restart of the scheduler has occurred due to a network error. Please investigate the network error to see if it can be avoided.
CTLW0015_MSG	An error occurred when putting the environment variable PROCESS_BATCH_NAME into the environment. The value of the variable is as follows:
CTLW0016_MSG	An error occurred when putting the environment variable PROCESS_GROUP_NAME into the environment. The value of the variable is as follows:
CTLW0800_MSG	MetaData mismatch detected. The data types of the source and target field do not match. The source and target fields are as follows:
CTLW0801_MSG	MetaData mismatch detected. The number of decimal digits of the source field is greater than the number of decimal digits of the target field. Right data truncation may occur. The source and target fields are as follows:
CTLS0001_MSG	Program Terminating.
CTLS0002_MSG	Could not connect to DBConnectionInParameter. Value of DBConnectionInParameter is as follows:
CTLS0003_MSG	The CTLF001FileName input parameter was not set. CTLAG01 requires that this parameter is set because it is the input file to the program.
CTLS0004_MSG	There was no data found in the input table InTable. The select statement used is as follows:
CTLS0005_MSG	There was an open error for the input table. Refer to ODBC Messages in the error message log. The select statement used is as follows:
CTLS0006_MSG	The number of result columns from the select from InTable is equal to zero. The select statement used is as follows:
CTLS0007_MSG	Could not open WorkFileName for writing. Value of WorkFileName is as follows:
CTLS0008_MSG	Could not prepare WorkFileName for writing. Value of WorkFileName is as follows:
CTLS0009_MSG	Could not write to WorkFileName. Value of WorkFileName is as follows:
CTLS0010_MSG	Could not connect to DBConnectionOutParameter. Value of DBConnectionOutParameter is as follows:
CTLS0011_MSG	There was an open error opening the output table. Refer to ODBC Messages in the error message log. The select statement used is as follows:
CTLS0012_MSG	The number of result columns from the validation of OutTable is equal to zero. The select statement used is as follows:
CTLS0013_MSG	The preparation for the insert statement for OutTable failed. The fully qualified name of the OutTable is as follows:

CTLS0014_MSG	The preparation for the update statement for OutTable failed. The fully qualified name of the OutTable is as follows:
CTLS0015_MSG	The preparation for the delete statement for OutTable failed. The fully qualified name of the OutTable is as follows:
CTLS0016_MSG	Could not open WorkFileName for reading. Value of WorkFileName is as follows:
CTLS0017_MSG	Could not prepare WorkFileName for reading. Value of WorkFileName is as follows:
CTLS0018_MSG	Could not read from WorkFileName. Value of WorkFileName is as follows:
CTLS0019_MSG	Error encountered when transferring data from WorkFileName to Insert Database pointer to perform insert. Contact technical support. The fully qualified table name for the insert table is as follows:
CTLS0020_MSG	The insert encountered a clash of primary keys or unique index and the program parameter specifies 'insert only' for rows. If you want key/index clashes to be resolved automatically change the InsertUpdateOption. The fully qualified table name for the insert table is as follows:
CTLS0021_MSG	Error encountered when transferring data from Insert database pointer to the Update database pointer to perform update. Contact technical support. The fully qualified table name for the update table is as follows:
CTLS0022_MSG	Error encountered when performing the update after a failed insert. Refer to ODBC Messages in the error message log. The fully qualified table name for the update table is as follows:
CTLS0023_MSG	Error encountered when performing the delete after a failed insert. Refer to ODBC Messages in the error message log. The fully qualified table name for the delete table is as follows:
CTLS0024_MSG	Error encountered when performing the insert after a failed insert. Refer to ODBC Messages in the error message log. The fully qualified table name for the insert table is as follows:
CTLS0025_MSG	Error encountered when transferring data from Insert database pointer to the Delete database pointer to perform delete. Contact technical support. The fully qualified table name for the delete table is as follows:
CTLS0026_MSG	Error encountered when transferring data from WorkFileName to Update database pointer to perform update. Contact technical support. The fully qualified table name for the update table is as follows:
CTLS0027_MSG	Error encountered when transferring data from Update database pointer to the Insert database pointer to perform insert. Contact technical support. The fully qualified table name for the insert table is as follows:
CTLS0028_MSG	Error encountered when performing the insert after a failed update. Refer to ODBC Messages in the error message log. The fully qualified table name for the insert table is as follows:
CTLS0029_MSG	Error encountered when performing update. Refer to ODBC Messages in the error message log. The fully qualified table name for the update table is as follows:
CTLS0030_MSG	The parameter defining the WorkFileName is not set. The WorkFileName must be set to a file that the program can open for reading and writing.
CTLS0031_MSG	Error encountered when performing the validate of the aggregation control table. Refer to ODBC Messages in the error message log. The select statement used to validate the aggregation control table is as follows:
CTLS0032_MSG	Error encountered when performing the open of the aggregation control table. Refer to ODBC Messages in the error message log. The select statement used to open the aggregation control table is as follows:
CTLS0033_MSG	An unexpected end of data was encountered when reading the aggregation control table. There must be at least one record in the aggregation control table specifying a summary level for the fact table specified in the input parameters. The select statement used to read rows from the aggregation control table is as follows:
CTLS0034_MSG	Could not open CTLF001FileName for reading. Value of CTLF001FileName is as follows:
CTLS0035_MSG	Could not prepare CTLF001FileName for reading. Value of CTLF001FileName is as follows:
CTLS0036_MSG	Memory allocation error. The program was unable to allocate memory to hold data in memory.
CTLS0037_MSG	Could not open CTLF002FileName for writing. Value of CTLF002FileName is as follows:
CTLS0038_MSG	Could not prepare CTLF002FileName for writing. Value of CTLF002FileName is as follows:
CTLS0039_MSG	Function f2010_initialise_for_aggregate_control_record_processing failed. Review Error Log for more messages.
CTLS0040_MSG	Function f2020_process_CTLF001_producing_sort_work_1 failed. Review Error Log for more messages.
CTLS0041_MSG	Function f2030_process_sort_work_1_producing_sort_work_2 failed. Review Error Log for

	more messages.
CTLS0042_MSG	Function f2040_process_sort_work_2_producing_CTLF002 failed. Review Error Log for more messages.
CTLS0043_MSG	Error encountered when performing the delete of all records in the sort work 1 table. Refer to ODBC Messages in the error message log. The sql executed to perform the delete is as follows:
CTLS0044_MSG	Error encountered when performing the delete of all records in the sort work 2 table. Refer to ODBC Messages in the error message log. The sql executed to perform the delete is as follows:
CTLS0045_MSG	The preparation for the insert statement for the sort work 1 table failed. Refer to ODBC Messages in the error message log. The sort work 1 table name is as follows:
CTLS0046_MSG	The file CTLF001FileName is empty. Program CTLAG01 should not be executed if there has been no data input to CTLAT01. Value of CTLF001FileName is as follows:
CTLS0047_MSG	Function f2090_send_data_to_sort_work_1 failed. Review Error Log for more messages.
CTLS0049_MSG	Error encountered when performing the insert into the sort work 1 table. Refer to ODBC Messages in the error message log. The sort work 1 table name is as follows:
CTLS0050_MSG	Error encountered when performing the insert select sort sum group by statement to summarise data from sort work table 1 and place it into sort work table 2. Refer to ODBC Messages in the error message log. The insert select statement executed is as follows:
CTLS0051_MSG	The close of the prepared insert statement for the sort work 1 table failed. Refer to ODBC Messages in the error message log. The sort work 1 table name is as follows:
CTLS0052_MSG	Error encountered when performing the open of the sort work 2 table. Refer to ODBC Messages in the error message log. The select statement used to open the sort work 2 table is as follows:
CTLS0053_MSG	Error encountered when performing the read of the sort work 2 table. Refer to ODBC Messages in the error message log. The select statement used to read the sort work 2 table is as follows:
CTLS0054_MSG	The CTLF001FileName input parameter was not set. CTLAT01 requires that this parameter is set because it is the output file from the program.
CTLS0055_MSG	Error encountered when performing the validate of the input table. Refer to ODBC Messages in the error message log. The select statement used to validate the input table is as follows:
CTLS0056_MSG	Error encountered when performing the open of the input table. Refer to ODBC Messages in the error message log. The select statement used to open the input table is as follows:
CTLS0057_MSG	No input data for CTLAT01 to process. This is defined to be an error. Please do not submit the fact table attribution process to run if there is no input data available
CTLS0058_MSG	Error encountered when performing the validate of a dimension table. Refer to ODBC Messages in the error message log. The select statement used to validate the dimension table is as follows:
CTLS0059_MSG	Could not open CTLF001FileName for writing. Value of CTLF001FileName is as follows:
CTLS0060_MSG	Could not prepare CTLF001FileName for writing. Value of CTLF001FileName is as follows:
CTLS0061_MSG	Function f3100_attribute_record failed. Review Error Log for more messages.
CTLS0062_MSG	Error encountered when writing CTLF001FileName. Value of CTLF001FileName is as follows:
CTLS0063_MSG	The CTLF002FileName input parameter was not set. CTLAG01 requires that this parameter is set because it is the output file from the program.
CTLS0064_MSG	The CTLF002FileName input parameter was not set. CTLCL01 requires that this parameter is set because it is the input file to the program.
CTLS0065_MSG	The CTLF003FileName input parameter was not set. CTLCL01 requires that this parameter is set because it is an output file from the program.
CTLS0066_MSG	The CTLF004FileName input parameter was not set. CTLCL01 requires that this parameter is set because it is an output file from the program.
CTLS0067_MSG	Error encountered when performing the validate of the summary fact table. Refer to ODBC Messages in the error message log. The select statement used to validate the summary fact table is as follows:
CTLS0068_MSG	Could not open CTLF002FileName for reading. Value of CTLF002FileName is as follows:
CTLS0069_MSG	Could not prepare CTLF002FileName for reading. Value of CTLF002FileName is as follows:
CTLS0070_MSG	The file CTLF002FileName is empty. Program CTLCL01 should not be executed if there has been no data input to CTLAT01. Value of CTLF002FileName is as follows:
CTLS0071_MSG	The preparation for the insert statement for the summary fact table failed. CTLAG01 prepares an insert statement to discover the characteristics of the summary fact table. The fully qualified name of the summary fact table is as follows:

CTLS0072_MSG	The preparation for the insert statement for the summary fact table failed. CTLCL01 prepares an insert statement to discover the characteristics of the summary fact table. The fully qualified name of the summary fact table is as follows:
CTLS0073_MSG	The preparation for the select statement for the summary fact table failed. Refer to ODBC Messages in the error message log. CTLCL01 prepares a select statement to read the summary fact table. The fully qualified name of the summary fact table is as follows:
CTLS0074_MSG	Could not open CTLF003FileName for writing. Value of CTLF003FileName is as follows:
CTLS0075_MSG	Could not prepare CTLF003FileName for writing. Value of CTLF003FileName is as follows:
CTLS0076_MSG	Could not open CTLF004FileName for writing. Value of CTLF004FileName is as follows:
CTLS0077_MSG	Could not prepare CTLF004FileName for writing. Value of CTLF004FileName is as follows:
CTLS0078_MSG	Error encountered when writing CTLF003FileName. Value of CTLF003FileName is as follows:
CTLS0079_MSG	Error encountered when writing CTLF004FileName. Value of CTLF004FileName is as follows:
CTLS0080_MSG	Error encountered when performing the open of the input table for the dimension table data. Refer to ODBC Messages in the error message log. The select statement used to open the input table is as follows:
CTLS0081_MSG	The number of columns returned by the select statement that opened the input table for the dimension table data returned zero columns. The select statement used to open the input table is as follows:
CTLS0082_MSG	Function f3090_update_aggregate_keys_control failed. Review Error Log for more messages.
CTLS0083_MSG	Error encountered when performing the validate of the input table for dimension table data. Refer to ODBC Messages in the error message log. The select statement used to validate the input table is as follows:
CTLS0084_MSG	Error encountered when performing the validate of the base type 2 dimension table. Refer to ODBC Messages in the error message log. The select statement used to validate the base dimension table is as follows:
CTLS0085_MSG	Error encountered when performing the validate of the output dimension table. Refer to ODBC Messages in the error message log. The select statement used to validate the output dimension table is as follows:
CTLS0086_MSG	Error encountered when performing the validate of the dimension table key definitions table. Refer to ODBC Messages in the error message log. The select statement used to validate the dimension table key definitions table is as follows:
CTLS0087_MSG	Error encountered when performing the validate of the dimension table type 2 column defintions table. Refer to ODBC Messages in the error message log. The select statement used to validate the dimension table type 2 column defintions table is as follows:
CTLS0088_MSG	Error encountered when performing the validate of the aggregate keys control table. Refer to ODBC Messages in the error message log. The select statement used to validate the aggregate keys control table is as follows:
CTLS0089_MSG	Error encountered when performing the open of the aggregate keys control table. Refer to ODBC Messages in the error message log. The select statement used to open the aggregate keys control table is as follows:
CTLS0090_MSG	Error encountered when performing the validate of the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw batch control table is as follows:
CTLS0091_MSG	Error encountered when performing the validate of the dw month control table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw month control table is as follows:
CTLS0092_MSG	Error encountered when performing the open of the dimension table key definitions table. Refer to ODBC Messages in the error message log. The select statement used to open the dimension table key definitions table is as follows:
CTLS0093_MSG	The level of aggregate in 'agg_level' field in the dimension table key definitions table exceeds the maximum number of levels of aggregates in SeETL[RT]. Please change the 'agg_level' field on the dimension table key definitions table.
CTLS0094_MSG	Error encountered when performing the open of the dimension table type 2 column definitions table. Refer to ODBC Messages in the error message log. The select statement used to open the dimension table type 2 column definitions table is as follows:
CTLS0095_MSG	The preparation for the select statement for the output dimension table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the output dimension table is as follows:

CTLS0096_MSG	Error encountered when performing the open of the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch control table table is as follows:
CTLS0097_MSG	Error encountered when performing the open of the dw month control table. Refer to ODBC Messages in the error message log. The select statement used to open the dw month control table table is as follows:
CTLS0098_MSG	The preparation for the insert statement for the output dimension table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the output dimension table is as follows:
CTLS0099_MSG	The preparation for the first update statement for the output dimension table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the output dimension table is as follows:
CTLS0100_MSG	The preparation for the second statement for the output dimension table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the output dimension table is as follows:
CTLS0101_MSG	Function f3030_get_integer_keys failed. Review Error Log for more messages.
CTLS0102_MSG	Function f3060_update_rows failed. Review Error Log for more messages.
CTLS0103_MSG	Function f3050_insert_rows failed. Review Error Log for more messages.
CTLS0104_MSG	Function f3061_close_dimension_record failed. Review Error Log for more messages.
CTLS0105_MSG	Function f3061_close_dimension_record failed. Review Error Log for more messages.
CTLS0106_MSG	Function f3062_update_dimension_record failed. Review Error Log for more messages.
CTLS0107_MSG	Error encountered when performing the prepared select on the output dimension table to look up the level keys. Refer to ODBC Messages in the error message log. The fully qualified table name of the output dimension table is as follows:
CTLS0108_MSG	Error encountered when performing the prepared insert on the output dimension table. Refer to ODBC Messages in the error message log. The fully qualified table name of the output dimension table is as follows:
CTLS0109_MSG	Error encountered when performing the prepared update on the output dimension table. Refer to ODBC Messages in the error message log. The fully qualified table name of the output dimension table is as follows:
CTLS0110_MSG	Error encountered when performing the prepared update to close out the row in the output dimension table. Refer to ODBC Messages in the error message log. The fully qualified table name of the output dimension table is as follows:
CTLS0111_MSG	Error encountered when performing the prepared update on the output dimension table. Refer to ODBC Messages in the error message log. The fully qualified table name of the output dimension table is as follows:
CTLS0112_MSG	The preparation for the update statement for the aggregate keys control table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the aggregate keys control table is as follows:
CTLS0113_MSG	Error encountered when performing the prepared update on the aggregate keys control table. Refer to ODBC Messages in the error message log. The fully qualified table name of the aggregate keys control table is as follows:
CTLS0114_MSG	Error encountered when performing the insert. Refer to ODBC Messages in the error message log. The fully qualified table name for the insert table is as follows:
CTLS0115_MSG	An attempt has been made to execute the program after the Evaluation Period has expired. Contact peter@peternolan.com to discuss obtaining a new evaluation copy of SeETL[RT].
CTLS0116_MSG	Error encountered when performing the validate of the dim table load control table. Refer to ODBC Messages in the error message log. The select statement used to validate the dim table load control table is as follows:
CTLS0117_MSG	Function f1000_initialise_part2 failed. Review Error Log for more messages.
CTLS0118_MSG	The dimension table load to memory function could not open the dimension table. Refer to ODBC Messages in the error message log. The select statement used to open the dimension table is as follows:
CTLS0119_MSG	The dimension table load to memory function determined that there are zero rows in the dimension table. Zero rows in a dimension table is not allowed. The select statement used to open the dimension table is as follows:
CTLS0120_MSG	The dimension table load to memory function determined that there are less than 2 columns returned by a select * statement against a dimension table. This is not allowed. The view of the dimension table must return at least the dimension table character key field and one integer key. The select statement used to open the dimension table is as follows:
CTLS0121_MSG	The dimension table load to memory function has been unable to allocate memory for the

	pointer to store the pointers to the dimension character keys(ptr_ptr_dim_char_ky_fld). More memory is needed, or turn off loading into memory for the dimension table. The select statement used to open the dimension table is as follows:
CTLS0122_MSG	The dimension table load to memory function has been unable to allocate memory to store the integer keys(ptr_ws_dim_int_keys). More memory is needed, or turn off loading into memory for the dimension table. The select statement used to open the dimension table is as follows:
CTLS0123_MSG	The dimension table load to memory function has been unable to allocate memory store a dimension character key(ptr_dim_char_ky_fld). More memory is needed, or turn off loading into memory for the dimension table. The select statement used to open the dimension table is as follows:
CTLS0124_MSG	The dimension table load to memory function encountered an unexecpted error. Refer to ODBC Messages in the error message log. The select statement used to open the dimension table is as follows:
CTLS0125_MSG	Function PrepareDimensionSelect() failed. Review Error Log for more messages.
CTLS0126_MSG	Function PerformPreparedDimensionSelect() failed. Review Error Log for more messages.
CTLS0127_MSG	Error encountered when performing the validate of the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw batch control table is as follows:
CTLS0128_MSG	Error encountered when performing the open of the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch control table is as follows:
CTLS0129_MSG	The dw batch control indicates that there is an incomplete batch. You may not start a new batch until the previous batch is completed. The select statement used to check the dw batch control table is as follows:
CTLS0130_MSG	Error encountered when performing the insert of the new dw batch control table record. Refer to ODBC Messages in the error message log. The insert statement used to insert the new dw batch control record is as follows:
CTLS0131_MSG	Error encountered when performing the update to close out the dw batch control table record. Refer to ODBC Messages in the error message log. The update statement used to close out the dw batch control record is as follows:
CTLS0132_MSG	Error encountered when performing an SQL Statement in CTLU002. Refer to ODBC Messages in the error message log. The SQL statement issued is as follows:
CTLS0133_MSG	The SQLFileName input parameter was not set. CTLU001 requires that this parameter is set because it is the input file to the program."
CTLS0134_MSG	Could not open SQLFileName for reading. Value of SQLFileName is as follows:
CTLS0135_MSG	Error encountered when performing the validate of the dw ddl gen table table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw ddl gen table table is as follows:
CTLS0136_MSG	Error encountered when performing the validate of the dw ddl gen columns table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw ddl gen columns table is as follows:
CTLS0137_MSG	Error encountered when performing the open of the dw ddl gen table table. Refer to ODBC Messages in the error message log. The select statement used to open the dw ddl gen table table is as follows:
CTLS0138_MSG	Function f1000_initialise_part1 failed. Review Error Log for more messages.
CTLS0139_MSG	Error encountered when performing the open of the dw ddl gen columns table. Refer to ODBC Messages in the error message log. The select statement used to open the dw ddl gen columns table is as follows:
CTLS0140_MSG	Could not open Output File for Table DDL for writing. Value of Output File for Table DDL is as follows:
CTLS0141_MSG	Could not open Output File for View DDL for writing. Value of Output File for View DDL is as follows:
CTLS0142_MSG	Error encountered when reading the dw batch control table to obtain a close date for the Type 2 dimension table. Refer to ODBC Messages in the error message log. The select statement used to read the close date from the dw batch control table table is as follows:
CTLS0143_MSG	Error encountered when performing the validate of the output profile table. Refer to ODBC Messages in the error message log. The select statement used to validate the profile table is as follows:
CTLS0144_MSG	The preparation for the select statement for the profile table failed. Refer to ODBC Messages in the error message log. CTLPR01 prepares a select statement to read the

	profile table. The fully qualified name of the profile table is as follows:
CTLS0145_MSG	Error encountered when performing the validate of the output update profile table. Refer to ODBC Messages in the error message log. The select statement used to validate the output update profile table is as follows:
CTLS0146_MSG	Could not open CTLF005FileName for writing. Value of CTLF005FileName is as follows:
CTLS0147_MSG	Could not prepare CTLF005FileName for writing. Value of CTLF005FileName is as follows:
CTLS0148_MSG	Error encountered when performing the open of the output profile table. Refer to ODBC Messages in the error message log. The select statement used to open the profile table is as follows:
CTLS0149_MSG	Could not open CTLF006FileName for writing. Value of CTLF006FileName is as follows:
CTLS0150_MSG	Could not prepare CTLF006FileName for writing. Value of CTLF006FileName is as follows:
CTLS0151_MSG	Error encountered when performing the open of the output for update of the profile table. Refer to ODBC Messages in the error message log. The select statement used to open the output for update for the profile table is as follows:
CTLS0152_MSG	Error encountered when performing the validate of the first lookup table for the profile table. Refer to ODBC Messages in the error message log. The select statement used to validate the first lookup table for the profile table is as follows:
CTLS0153_MSG	Error encountered when performing the validate of the insert for the profile table. Refer to ODBC Messages in the error message log. The select statement used to validate the insert for the profile table is as follows:
CTLS0154_MSG	The CTLF005FileName input parameter was not set. CTLPR01 requires that this parameter is set because it is an output file from the program.
CTLS0155_MSG	The CTLF006FileName input parameter was not set. CTLPR01 requires that this parameter is set because it is an output file from the program.
CTLS0156_MSG	Error encountered when performing the validate of the second lookup table for the profile table. Refer to ODBC Messages in the error message log. The select statement used to validate the second lookup table for the profile table is as follows:
CTLS0157_MSG	Error encountered when performing the validate of the time dim prfl lkp table for the profile table. This table is required as it supplies the keys for the date from date to values on the profile table. Refer to ODBC Messages in the error message log. The select statement used to validate the time dim prfl lkp table for the profile table is as follows:
CTLS0158_MSG	Error encountered when reading the dw batch control table to obtain a close date for the profile table. Refer to ODBC Messages in the error message log. The select statement used to read the close date from the dw batch control table table is as follows:
CTLS0159_MSG	Error encountered when reading the dw batch control table to obtain an open date for the profile table. Refer to ODBC Messages in the error message log. The select statement used to read the open date from the dw batch control table table is as follows:
CTLS0160_MSG	Error encountered when performing the prepare for the select from the time dim prfl lkp table for the profile table. Refer to ODBC Messages in the error message log. The fully qualified name of the time dim prfl lkp table is as follows:
CTLS0161_MSG	Error encountered when performing the select from the time dim prfl lkp table for the profile table. Refer to ODBC Messages in the error message log. The fully qualified name of the time dim prfl lkp table is as follows:
CTLS0162_MSG	The preparation for the select statement for the first lookup table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the first lookup table is as follows:
CTLS0163_MSG	The preparation for the select statement for the second lookup table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the second lookup table is as follows:
CTLS0164_MSG	Error encountered when performing the open of the insert view for the profile table. Refer to ODBC Messages in the error message log. The select statement used to open the insert view of the profile table is as follows:
CTLS0165_MSG	No input data for CTLPR01 to process. This is defined to be an error. It is not expected that any profiling process will have no input. Please do not submit the profiling process to run if there is no input data available.
CTLS0166_MSG	Function f3200_attribute_record failed. Review Error Log for more messages.
CTLS0167_MSG	Error encountered when performing the select from the first lookup table for the profile table. Refer to ODBC Messages in the error message log. The fully qualified name of the first lookup table is as follows:
CTLS0168_MSG	Error encountered when writing CTLF005FileName. Value of CTLF005FileName is as follows:

CTLS0169_MSG	Error encountered when performing the select from the second lookup table for the profile table. Refer to ODBC Messages in the error message log. The fully qualified name of the second lookup table is as follows:
CTLS0170_MSG	Error encountered when writing CTLF006FileName. Value of CTLF006FileName is as follows:
CTLS0171_MSG	Function PrepareProfileSelect() failed. Review Error Log for more messages.
CTLS0172_MSG	Error encountered when writing CTLF007FileName. Value of CTLF007FileName is as follows:
CTLS0173_MSG	The CTLF007FileName input parameter was not set. CTLAT02 requires that this parameter is set because it is an output file from the program.
CTLS0174_MSG	No input data for CTLAT02 to process. This is defined to be an error. Please do not submit the fact table attribution process to run if there is no input data available.
CTLS0175_MSG	The number of columns returned by the select statement that opens the dimension table data returned zero columns. The select statement used to open the table is as follows:
CTLS0176_MSG	The number of columns returned by the select statement that opens the lookup profile table data returned zero columns. The select statement used to open the lookup for the profile table is as follows:
CTLS0177_MSG	Could not open CTLF007FileName for writing. Value of CTLF007FileName is as follows:
CTLS0178_MSG	Could not prepare CTLF007FileName for writing. Value of CTLF007FileName is as follows:
CTLS0179_MSG	Function PerformPreparedProfileSelect() failed. Review Error Log for more messages.
CTLS0180_MSG	The CTLF007FileName input parameter was not set. CTLAS01 requires that this parameter is set because it is an output file from the program.
CTLS0181_MSG	The CTLF008FileName input parameter was not set. CTLAS01 requires that this parameter is set because it is an output file from the program.
CTLS0182_MSG	Error encountered when performing the validate of the output association table. Refer to ODBC Messages in the error message log. The select statement used to validate the association table is as follows:
CTLS0183_MSG	Error encountered when performing the validate of the output update association table. Refer to ODBC Messages in the error message log. The select statement used to validate the output update association table is as follows:
CTLS0184_MSG	Error encountered when performing the validate of the first lookup table for the association table. Refer to ODBC Messages in the error message log. The select statement used to validate the first lookup table for the association table is as follows:
CTLS0185_MSG	Error encountered when performing the validate of the second lookup table for the association table. Refer to ODBC Messages in the error message log. The select statement used to validate the second lookup table for the association table is as follows:
CTLS0186_MSG	Error encountered when performing the validate of the insert for the association table. Refer to ODBC Messages in the error message log. The select statement used to validate the insert for the association table is as follows:
CTLS0187_MSG	Error encountered when performing the validate of the time dim asoc lkp table for the association table. This table is required as it supplies the keys for the date from date to values on the association table. Refer to ODBC Messages in the error message log. The select statement used to validate the time dim asoc lkp table for the association table is as follows:
CTLS0188_MSG	Error encountered when reading the dw batch control table to obtain a close date for the association table. Refer to ODBC Messages in the error message log. The select statement used to read the close date from the dw batch control table table is as follows:
CTLS0189_MSG	Error encountered when reading the dw batch control table to obtain an open date for the association table. Refer to ODBC Messages in the error message log. The select statement used to read the open date from the dw batch control table table is as follows:
CTLS0190_MSG	Error encountered when performing the prepare for the select from the time dim asoc lkp table for the association table. Refer to ODBC Messages in the error message log. The fully qualified name of the time dim asoc lkp table is as follows:
CTLS0191_MSG	Error encountered when performing the select from the time dim asoc lkp table for the association table. Refer to ODBC Messages in the error message log. The fully qualified name of the time dim asoc lkp table is as follows:
CTLS0192_MSG	The preparation for the select statement for the first lookup table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the first lookup table is as follows:
CTLS0193_MSG	The preparation for the select statement for the second lookup table failed. Refer to ODBC Messages in the error message log. The fully qualified name of the second lookup table is as

	follows:
CTLS0194_MSG	Error encountered when performing the open of the insert view for the association table. Refer to ODBC Messages in the error message log. The select statement used to open the insert view of the association table is as follows:
CTLS0195_MSG	Error encountered when performing the open of the output for update of the association table. Refer to ODBC Messages in the error message log. The select statement used to open the output for update for the association table is as follows:
CTLS0196_MSG	No input data for CTLAS01 to process. This is defined to be an error. It is not expected that any association process will have no input. Please do not submit the association process to run if there is no input data available.
CTLS0197_MSG	Could not open CTLF007FileName for writing. Value of CTLF007FileName is as follows:
CTLS0198_MSG	Could not prepare CTLF007FileName for writing. Value of CTLF007FileName is as follows:
CTLS0199_MSG	Could not open CTLF008FileName for writing. Value of CTLF008FileName is as follows:
CTLS0200_MSG	Could not prepare CTLF008FileName for writing. Value of CTLF008FileName is as follows:
CTLS0201_MSG	Error encountered when performing the select from the first lookup table for the association table. Refer to ODBC Messages in the error message log. The fully qualified name of the first lookup table is as follows:
CTLS0202_MSG	Error encountered when writing CTLF007FileName. Value of CTLF007FileName is as follows:
CTLS0203_MSG	Error encountered when performing the select from the second lookup table for the association table. Refer to ODBC Messages in the error message log. The fully qualified name of the second lookup table is as follows:
CTLS0204_MSG	Error encountered when writing CTLF008FileName. Value of CTLF008FileName is as follows:
CTLS0205_MSG	The CTLF001FileName input parameter was not set. CTLU005 requires that this parameter is set because it is the input file to the program.
CTLS0206_MSG	The CTLF002FileName input parameter was not set. CTLU005 requires that this parameter is set because it is the output file from the program.
CTLS0207_MSG	The parameters MoveByColumnName and MoveByColumnPosition are both set to Yes. This is not allowed. This may be because of the defaulting behavior of the parameters. You may Move By ColumnName or Move By ColumnPosition. Please set the parameters to this program so that only one of these parameters is set to Yes.
CTLS0208_MSG	The parameters MoveByColumnName and MoveByColumnPosition are both set to No. This is not allowed. This may be because of the defaulting behavior of the parameters. You may Move By ColumnName or Move By ColumnPosition. Please set the parameters to this program so that one of these parameters is set to Yes.
CTLS0209_MSG	The parameter HeaderRecordExists is set to No and the parameter MoveByColumnName is set to Yes. This is not allowed since there are no column names provided in a header record to be able to move the columns of data by column name. If there is no header record in the input file you must set MoveByColumnName to No and MoveByColumnPosition to Yes.
CTLS0210_MSG	Program CTLU005 uses an output table to create the DTU self describing file format for the output file. An error was encountered when performing the validate of the output table. Refer to ODBC Messages in the error message log. The select statement used to validate the output table is as follows:
CTLS0211_MSG	Program CTLU005 uses an output table to create the DTU self describing file format for the output file. An error was encountered when performing the open of the output table. Refer to ODBC Messages in the error message log. The select statement used to open the output table is as follows:
CTLS0212_MSG	Could not open CTLF001FileName for reading. Value of CTLF001FileName is as follows:
CTLS0213_MSG	Could not open CTLF002FileName for writing. Value of CTLF002FileName is as follows:
CTLS0214_MSG	Could not prepare CTLF001FileName for reading. Value of CTLF001FileName is as follows:
CTLS0215_MSG	An error was encountered when reading from CTLF001FileName. Value of CTLF001FileName is as follows:
CTLS0216_MSG	An error was encountered when writing to CTLF002FileName. Value of CTLF002FileName is as follows:
CTLS0217_MSG	The CTLF001FileName input parameter was not set. CTLU006 requires that this parameter is set because it is the input file to the program.
CTLS0218_MSG	The CTLF002FileName input parameter was not set. CTLU006 requires that this parameter is set because it is the output file from the program.
CTLS0219_MSG	The CTLF001FileNameFormat input parameter was not set. CTLU006 requires that this parameter is set because it contains the format definitions for the fixed format input file.

CTLS0220_MSG	The parameter defining the LoadImageFileName is not set. The LoadImageFileName must be set to a file that the program can open for writing.
CTLS0221_MSG	Could not open LoadImageFileName for reading. Value of LoadImageFileName is as follows:
CTLS0222_MSG	Program CTLU001 uses an output table to create the format of Load Image File. An error was encountered when performing the validate of the output table. Refer to ODBC Messages in the error message log. The select statement used to validate the output table is as follows:
CTLS0223_MSG	Program CTLU001 uses an output table to create the format of Load Image File. An error was encountered when performing the open of the output table. Refer to ODBC Messages in the error message log. The select statement used to open the output table is as follows:
CTLS0224_MSG	Could not open LoadImageFileName for writing. Value of LoadImageFileName is as follows:
CTLS0225_MSG	Could not prepare LoadImageFileName for writing. Value of LoadImageFileName is as follows:
CTLS0226_MSG	Error encountered when performing the delete of a record found in both the WorkFileName and the OutTable. This record should be deleted by CTLU001 and then loaded by the database loader. Refer to ODBC Messages in the error message log. The fully qualified table name for the delete table is as follows:
CTLS0227_MSG	An error was encountered when writing to LoadImageFileName. Value of LoadImageFileName is as follows:
CTLS0228_MSG	Program CTLU006 uses an output table to create the DTU self describing file format for the output file. An error was encountered when performing the validate of the output table. Refer to ODBC Messages in the error message log. The select statement used to validate the output table is as follows:
CTLS0229_MSG	Program CTLU006 uses an output table to create the DTU self describing file format for the output file. An error was encountered when performing the open of the output table. Refer to ODBC Messages in the error message log. The select statement used to open the output table is as follows:
CTLS0230_MSG	Could not open CTLF001FileNameFormat for reading. Value of CTLF001FileNameFormat is as follows:
CTLS0231_MSG	Could not prepare CTLF001FileNameFormat for reading. Value of CTLF001FileNameFormat is as follows:
CTLS0232_MSG	An error was encountered when reading the next Fixed File Format record from CTLF001FileName. Value of CTLF001FileName is as follows:
CTLS0233_MSG	The CTLF001FileName input parameter was not set. CTLU007 requires that this parameter is set because it is an input file to the program.
CTLS0234_MSG	The CTLF002FileName input parameter was not set. CTLU007 requires that this parameter is set because it is an input file to the program.
CTLS0235_MSG	The CTLF003FileName input parameter was not set. CTLU007 requires that this parameter is set because it is the output file from the program.
CTLS0236_MSG	Program CTLU007 uses an output table to create the DTU self describing file format for the output file. An error was encountered when performing the validate of the output table. Refer to ODBC Messages in the error message log. The select statement used to validate the output table is as follows:
CTLS0237_MSG	Program CTLU007 uses an output table to create the DTU self describing file format for the output file. An error was encountered when performing the open of the output table. Refer to ODBC Messages in the error message log. The select statement used to open the output table is as follows:
CTLS0238_MSG	Could not open CTLF001FileName for reading. Value of CTLF001FileName is as follows:
CTLS0239_MSG	Could not prepare CTLF001FileName for reading. Value of CTLF001FileName is as follows:
CTLS0240_MSG	Could not open CTLF002FileName for reading. Value of CTLF002FileName is as follows:
CTLS0241_MSG	Could not prepare CTLF002FileName for reading. Value of CTLF002FileName is as follows:
CTLS0242_MSG	Could not open CTLF003FileName for writing. Value of CTLF003FileName is as follows:
CTLS0243_MSG	Could not prepare CTLF003FileName for writing. Value of CTLF003FileName is as follows:
CTLS0244_MSG	An error was encountered when reading the next record from CTLF001FileName. Value of CTLF001FileName is as follows:
CTLS0245_MSG	An error was encountered when reading the next record from CTLF002FileName. Value of CTLF002FileName is as follows:
CTLS0246_MSG	An error occurred when preparing the CTLF001PrimaryKeyString. The value of the CTLF001PrimaryKeyString is as follows:
CTLS0247_MSG	An error occurred when preparing the CTLF002PrimaryKeyString. The value of the CTLF002PrimaryKeyString is as follows:
CTLS0248_MSG	An error was encountered when writing the next record to CTLF003FileName. Value of

	CTLF003FileName is as follows:
CTLS0249_MSG	An error was encountered when reading the next record from CTLF001FileName. Value of CTLF001FileName is as follows:
CTLS0250_MSG	An error was encountered when reading the next record from CTLF002FileName. Value of CTLF002FileName is as follows:
CTLS0251_MSG	An error was encountered when writing the next record to CTLF003FileName. Value of CTLF003FileName is as follows:
CTLS0252_MSG	An error was encountered when reading the next record from CTLF001FileName. Value of CTLF001FileName is as follows:
CTLS0253_MSG	An error was encountered when writing the next record to CTLF003FileName. Value of CTLF003FileName is as follows:
CTLS0254_MSG	An error was encountered when reading the next record from CTLF002FileName. Value of CTLF002FileName is as follows:
CTLS0255_MSG	An error was encountered when writing the next record to CTLF003FileName. Value of CTLF003FileName is as follows:
CTLS0256_MSG	An error was encountered when reading the next record from CTLF002FileName. Value of CTLF002FileName is as follows:
CTLS0257_MSG	An error was encountered when writing the next record to CTLF003FileName. Value of CTLF003FileName is as follows:
CTLS0258_MSG	An error was encountered when reading the next record from CTLF001FileName. Value of CTLF001FileName is as follows:
CTLS0259_MSG	An error was encountered when transfering column data by column name between CTLF001FileName and CTLF002FileName. The values of CTLF001FileName and CTLF002FileName are as follows:"
CTLS0260_MSG	An error was encountered when transferring column data by column name between CTLF002FileName and CTLF003FileName. The values of CTLF002FileName and CTLF003FileName are as follows:
CTLS0261_MSG	An error was encountered when transferring column data by column name between CTLF001FileName and CTLF003FileName. The values of CTLF001FileName and CTLF003FileName are as follows:
CTLS0262_MSG	An error was encountered when transferring column data by column name between CTLF002FileName and CTLF003FileName. The values of CTLF002FileName and CTLF003FileName are as follows:
CTLS0263_MSG	An error was encountered when transferring column data by column name between CTLF002FileName and CTLF003FileName. The values of CTLF002FileName and CTLF003FileName are as follows:
CTLS0264_MSG	An error was encountered when transferring column data by column name between CTLF001FileName and CTLF003FileName. The values of CTLF001FileName and CTLF003FileName are as follows:
CTLS0265_MSG	An error was encountered when creating the CTLF001FileName output memory area. Value of CTLF001FileName is as follows:
CTLS0266_MSG	An error was encountered when creating the CTLF007FileName output memory area. Value of CTLF007FileName is as follows:
CTLS0267_MSG	The consolidation program CTLCL01 found a field data type that could not be converted to a numeric type in order to be consolidated. The field was moved but not consolidated. Incorrect results will have been produced on the consolidated fact table output file CTLF004FileName. Please check summary fact table field data types. Value of CTLF004FileName is as follows:
CTLS0268_MSG	An error was encountered when transferring column data by column name between WorkFileName and LoadImageFileName. The values of WorkFileName and LoadImageFileName are as follows:
CTLS0269_MSG	The function f2000_process_batch_schedule issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0270_MSG	Error encountered when performing the validate of the dw commands table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw commands table is as follows:
CTLS0271_MSG	Error encountered when performing the validate of the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw batch control table is as follows:
CTLS0272_MSG	Error encountered when performing the validate of the dw batch pre req table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw batch pre req table is as follows:

CTLS0273_MSG	Error encountered when performing the validate of the dw batch run log table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw batch run log table is as follows:
CTLS0274_MSG	Error encountered when performing the validate of the dw process run log table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw process run log table is as follows:
CTLS0275_MSG	Error encountered when performing the prepare for the insert into the dw batch run log table. Refer to ODBC Messages in the error message log. The fully qualified name of the table used to prepare the dw batch run log table for inserting is as follows:
CTLS0276_MSG	Error encountered when performing the prepare for the update to the dw batch run log table. Refer to ODBC Messages in the error message log. The fully qualified name of the table used to prepare the dw batch run log table for updating is as follows:
CTLS0277_MSG	The function f2010_wait_batch_complete_flag_off issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0278_MSG	Error encountered when performing the open for the dw batch pre req table. Refer to ODBC Messages in the error message log. One of the causes of this error is that the dw batch pre req table is empty. This is considered an error. It is not possible to schedule a batch if no batch exists. The select statement used to open the dw batch pre req table is as follows:
CTLS0279_MSG	The function f3010_check_batch_processing_status issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0280_MSG	The function f2020_check_batch_running_or_abended issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0281_MSG	The function f2030_check_batch_pre_reqs issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0282_MSG	The function f3000_start_batch_processing issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0283_MSG	Error encountered when performing the open for the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch control table is as follows:
CTLS0284_MSG	Error encountered when performing the open for the dw batch run log table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch run log table is as follows:
CTLS0285_MSG	Error encountered when performing the open for the dw batch run log table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch run log table is as follows:
CTLS0286_MSG	Error encountered when performing the insert into the dw batch run log table. Refer to ODBC Messages in the error message log. The dw batch run log table name is as follows:
CTLS0287_MSG	Error encountered when performing the open for the dw batch pre req table for query 2. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch pre req table for query 2 is as follows:
CTLS0288_MSG	Error encountered when performing the open for the dw batch pre req table for query 2. The error is that there is no pre requisite defined for the batch. This is an error. All batches must have some form of pre-requisite defined for them. The select statement used to open the dw batch pre req table for query 2 is as follows:
CTLS0289_MSG	The function f2040_check_file_exists issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0290_MSG	The function f2050_check_daily_pre_req issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0291_MSG	Error encountered when performing the open for the dw commands batch proc grp asn table. This is a view into the dw commands table to determine the relationship between batches and process groups. Refer to ODBC Messages in the error message log. The select statement used to open the dw commands batch proc grp asn table is as follows:
CTLS0292_MSG	Error encountered when performing the open for the dw commands batch proc grp asn table. This is a view into the dw commands table to determine the relationship between batches and process groups. The error is that there were no process groups found for the batch. This is an error. It is not possible to create a batch without creating a process group and a process within the batch. Please correct the definition of the processes to be run in the batch. The select statement used to open the dw commands batch proc grp asn table is as follows:
CTLS0293_MSG	Error encountered when performing the open for the dw batch run log table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch run log

	table is as follows:
CTLS0294_MSG	Error encountered when performing the open for the dw batch run log table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch run log table is as follows:
CTLS0295_MSG	Error encountered when performing the open for the dw commands table. Refer to ODBC Messages in the error message log. The select statement used to open the dw commands table is as follows:
CTLS0296_MSG	Error encountered when performing the open for the dw proc run log table. Refer to ODBC Messages in the error message log. The select statement used to open the dw proc run log table is as follows:
CTLS0297_MSG	Error encountered when performing the open for the dw batch run log table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch run log table is as follows:
CTLS0298_MSG	Error encountered when performing the update of the dw batch run log table. Refer to ODBC Messages in the error message log. The dw batch run log table name is as follows:
CTLS0299_MSG	Error encountered when performing the open for the dw proc run log table. Refer to ODBC Messages in the error message log. The select statement used to open the dw proc run log table is as follows:
CTLS0300_MSG	Error encountered when performing the open for the dw batch run log table. Refer to ODBC Messages in the error message log. The select statement used to open the dw batch run log table is as follows:
CTLS0301_MSG	Error encountered when performing the update of the dw batch run log table. Refer to ODBC Messages in the error message log. The dw batch run log table name is as follows:
CTLS0302_MSG	The function f2000_wait_process_group_pre_reqs issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0303_MSG	Error encountered when performing the open for the dw commands table. Refer to ODBC Messages in the error message log. The select statement used to open the dw commands table is as follows:
CTLS0304_MSG	The function f3000_process_command_record issued an error return code. Review the error message log for more information as to the cause of this error code. The select statement used to open the dw commands table is as follows:
CTLS0305_MSG	Error encountered when performing an insert into the dw proc grp run log table. Refer to ODBC Messages in the error message log. The insert is an effort to log the fact that the process group failed to execute in the dw proc grp run log table. The dw proc grp run log table name is as follows:
CTLS0306_MSG	Error encountered when performing an insert into the dw proc grp run log table. Refer to ODBC Messages in the error message log. The insert is an effort to log the fact that the process group executed successfully in the dw proc grp run log table. The dw proc grp run log table name is as follows:
CTLS0307_MSG	The parameter defining the ProcessBatchName is not set. The ProcessBatchName must be set to a batch that is defined in the dw batch pre req table. It is not valid to attempt to start a batch that does not exist in the dw batch pre req table.
CTLS0308_MSG	The parameter defining the ProcessGroupName is not set. The ProcessGroupName must be set to a process group that is defined in the dw commands table. It is not valid to attempt to start a process group that does not exist in the dw commands table.
CTLS0309_MSG	Error encountered when performing the validate of the dw commands table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw commands table is as follows:
CTLS0310_MSG	Error encountered when performing the validate of the dw batch control table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw batch control table is as follows:
CTLS0311_MSG	Error encountered when performing the validate of the dw proc grp pre req table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw proc grp pre req table is as follows:
CTLS0312_MSG	Error encountered when performing the validate of the dw proc grp run log table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw proc grp run log table is as follows:
CTLS0313_MSG	Error encountered when performing the validate of the dw proc run log table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw proc run log table is as follows:
CTLS0314_MSG	Error encountered when performing the open for the dw batch control table. Refer to ODBC

	Messages in the error message log. The select statement used to open the dw batch control table is as follows:
CTLS0315_MSG	Error encountered when performing the open for the dw batch control table. The cause of this error is that the CTLU009 program requires an overall batch to be in progress as recorded in the dw batch control table prior to submitting a batch of process groups to run. The dw batch control table acts a semaphor to stop any accidental submissions of batches. It is not valid to try and start a batch while the dw batch control table indicates that no batches are allowed to be started. The select statement used to open the dw batch control table is as follows:
CTLS0316_MSG	Error encountered when preparing the insert for the dw proc grp run log table. Refer to ODBC Messages in the error message log. The dw proc grp run log table name is as follows:
CTLS0317_MSG	Error encountered when preparing the insert for the dw proc run log table. Refer to ODBC Messages in the error message log. The dw proc run log table name is as follows:
CTLS0318_MSG	Error encountered when performing the open for the dw proc grp pre req table. Refer to ODBC Messages in the error message log. The select statement used to open the dw proc grp pre req table is as follows:
CTLS0319_MSG	The function f2010_wait_for_file issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0320_MSG	The function f2020_wait_for_process_group issued an error return code. Review the error message log for more information as to the cause of this error code.
CTLS0321_MSG	Error encountered when performing the open for the dw proc grp run log table. Refer to ODBC Messages in the error message log. The select statement used to open the dw proc grp run log table is as follows:
CTLS0322_MSG	Error encountered when performing the open for the dw proc run log table. Refer to ODBC Messages in the error message log. The select statement used to open the dw proc run log table is as follows:
CTLS0323_MSG	Error encountered when performing an insert into the dw proc run log table. Refer to ODBC Messages in the error message log. The insert is an effort to log the fact that the process executed successfully in the dw proc run log table. The dw proc run log table name is as follows:
CTLS0324_MSG	Error encountered when performing an insert into the dw proc run log table. Refer to ODBC Messages in the error message log. The insert is an effort to log the fact that the process did not execute successfully in the dw proc run log table. The dw proc run log table name is as follows:
CTLS0325_MSG	Error encountered when executing a command from the dw command table. Refer to ODBC Messages in the error message log. The command may have written error messages to the error message log. The command that was issued to the system is as follows:
CTLS0326_MSG	Function f2000_run_datastage_job failed. Review Error Log for more messages.
CTLS0327_MSG	The parameter defining the DataStageServerName is not set. The DataStageServerName must be set to the name of a DataStage Server that can be accessed from the machine that CTLU010 is running on. It is not valid to attempt to run a DataStage job without specifying the server that the job will run on.
CTLS0328_MSG	The parameter defining the DataStageProjectName is not set. The DataStageProjectName must be set to the name of the DataStage Project in which the DataStage job that is to be run is stored. It is not valid to attempt to run a DataStage job without specifying the project that the job is in.
CTLS0329_MSG	The parameter defining the DataStageUserName is not set. The DataStageUserName must be set to the name of a user that can access the DataStage Project in which the DataStage job that is to be run is stored. It is not valid to attempt to run a DataStage job without specifying the DataStage userid that will run the job.
CTLS0330_MSG	The parameter defining the DataStageUserPassword is not set. The DataStageUserPassword must be set to the password of the DataStageUserName so that the program can access the DataStage Project in which the DataStage job that is to be run is stored. It is not valid to attempt to run a DataStage job without specifying the password of the DataStage userid that will run the job.
CTLS0331_MSG	The parameter defining the DataStageJobName is not set. The DataStageJobName must be set to the name of the job that is to be run. It is not valid to attempt to run a DataStage job without specifying the name of the job.
CTLS0332_MSG	An error occurred when attempting to connect to the DataStage Project. The DataStage Server, Project, Job, Userid and error messages are as follows:
CTLS0333_MSG	An error occurred when attempting to open the job to be run in the DataStage Project. The DataStage Server, Project, Job, Userid and error messages are as follows:

CTLS0334_MSG	An error occurred when attempting to lock the job to be run in the DataStage Project. The DataStage Server, Project, Job, Userid and error messages are as follows:
CTLS0335_MSG	An error occurred when attempting to run the job in the DataStage Project. The DataStage Server, Project, Job, Userid and error messages are as follows:
CTLS0336_MSG	An error occurred when attempting to wait for the job to be run in the DataStage Project. The DataStage Server, Project, Job, Userid and error messages are as follows:
CTLS0337_MSG	An error occurred when attempting to start the CTLU009 program from inside CTLU008. The error message indicates that the argument list exceeds 1024 bytes. The parameters passed to CTLU009 are as follows:
CTLS0338_MSG	An error occurred when attempting to start the CTLU009 program from inside CTLU008. The error message indicates that the mode argument is invalid. The parameters passed to CTLU009 are as follows:
CTLS0339_MSG	An error occurred when attempting to start the CTLU009 program from inside CTLU008. The error message indicates that the file or path is not found. The parameters passed to CTLU009 are as follows:
CTLS0340_MSG	An error occurred when attempting to start the CTLU009 program from inside CTLU008. The error message indicates that the specified file is not executable or has invalid executable-file format. The parameters passed to CTLU009 are as follows:
CTLS0341_MSG	An error occurred when attempting to start the CTLU009 program from inside CTLU008. The error message indicates that not enough memory is available to execute new process. The parameters passed to CTLU009 are as follows:
CTLS0342_MSG	An error occurred when attempting to find the field called delta_ind on the output table for CTLU007. The output table must contain a field called delta_ind so that the program can write the delta indicator to the correct column on the table. To have an output table with no field called delta_ind is considered an error. Please correct and re-submit the program for running. The select statement used to select all columns from the output table is as follows:
CTLS0343_MSG	WorkFileName is empty but the parameter InputTableOrFileCanBeEmpty is not set to Y. It is invalid for a file to be empty if InputTableOrFileCanBeEmpty is not set to Y. Value of WorkFileName is as follows:
CTLS0344_MSG	CTLF001FileName is empty but the parameter InputTableOrFileCanBeEmpty is not set to Yes. It is invalid for a file to be empty if InputTableOrFileCanBeEmpty is not set to Yes. Value of CTLF001FileName is as follows:
CTLS0345_MSG	CTLF002FileName is empty but the parameter InputTableOrFileCanBeEmpty is not set to Yes. It is invalid for a file to be empty if InputTableOrFileCanBeEmpty is not set to Yes. Value of CTLF002FileName is as follows:
CTLS0346_MSG	An error occurred when attempting to set a parameter for the job to be run in the DataStage Project. The DataStage Server, Project, Job, Userid, Parameter Name, Parameter Value and error messages are as follows:
CTLS0347_MSG	Error encountered when performing the validate of the DataStage parameter table. Refer to ODBC Messages in the error message log. The select statement used to validate the DataStage parameter table is as follows:
CTLS0348_MSG	Error encountered when performing the open of the DataStage parameter table. Refer to ODBC Messages in the error message log. The select statement used to open the DataStage parameter table is as follows:
CTLS0349_MSG	The open of the DataStage parameter table returned less than 3 columns. The DataStage parameter table must contain at least 3 columns. One for an integer key, and then one for the parameter name and one for the parameter value. Please check the DataStage parameter table is correct. The select statement used to open the DataStage parameter table is as follows:
CTLS0350_MSG	The parameter defining the DataStageParameterTable is not set. The DataStageParameterTable must be set to pass parameters to the job that is to be run. It is not valid to attempt to run a DataStage job without specifying the name of the parameter table to run the job.
CTLS0351_MSG	An error occurred when attempting to set the warning message limit for the job to be run in the DataStage Project. The DataStage Server, Project, Job, Userid and error messages are as follows:
CTLS0352_MSG	An error occurred when attempting to set the row limit for the job to be run in the DataStage Project. The DataStage Server, Project, Job, Userid and error messages are as follows:
CTLS0353_MSG	The number of result columns from the validation of OutTable is less than 10. The program CTLU015 requries that the target table contain at least 10 columns. The select statement used to detect the number of columns in the OutTable is as follows:

CTLS0354_MSG	Error encountered when performing the insert. Refer to ODBC Messages in the error message log. The fully qualified table name for the insert table is as follows:
CTLS0355_MSG	Error encountered when performing the open of the dimension table load control table. Refer to ODBC Messages in the error message log. The select statement used to open the input table is as follows:
CTLS0356_MSG	The dimension table load to memory function has been unable to create the file to provide the physical storage for the memory mapped file. This is an unusual error. Please refer the error to your windows systems administrators to see if they can determine the cause of the failure of the CreateFile call. The file name used to perform the CreateFile is as follows:
CTLS0357_MSG	The parameter defining the KillFileName is not set. The KillFileName must be set to a file that the program can detect within the operating system.
CTLS0358_MSG	The parameter defining the WaitFileName is not set. The WaitFileName must be set to a file that the program can open for reading and writing.
CTLS0359_MSG	Could not open WaitFileName for writing. Value of WaitFileName is as follows:
CTLS0360_MSG	The TranslateDateFormat parameter is set to 'Yes' however the DateFormatFrom parameter is invalid. Please check the valid values of DateFormatFrom in the documentation. The value of the DateFormatFrom parameter is as follows:
CTLS0361_MSG	The TranslateDateTimeFormat parameter is set to 'Yes' however the DateTimeFormatFrom parameter is invalid. Please check the valid values of DateTimeFormatFrom in the documentation. The value of the DateTimeFormatFrom parameter is as follows:
CTLS0362_MSG	Program CTLU016 uses an output table to determine the data types of the fields that it is reformatting or correcting. An error was encountered when performing the validate of the output table. Refer to ODBC Messages in the error message log. The select statement used to validate the output table is as follows:
CTLS0363_MSG	Program CTLU016 uses an output table to determine the data types of the fields that it is reformatting or correcting. An error was encountered when performing the open of the output table. Refer to ODBC Messages in the error message log. The select statement used to open the output table is as follows:
CTLS0364_MSG	Could not open CTLF002FileName for writing. Value of CTLF002FileName is as follows:
CTLS0365_MSG	Could not prepare CTLF002FileName for writing. Value of CTLF002FileName is as follows:
CTLS0366_MSG	An error was encountered when writing the next record to CTLF002FileName. Value of CTLF002FileName is as follows:
CTLS0367_MSG	The unix based dimension table load to memory function has been unable to open a file representing the mutex for sharing memory for a dimension table. The file name used to perform the open is as follows:
CTLS0368_MSG	The unix based dimension table load to memory function has been unable to open a file representing the semaphore that defines if a dimension table has been created for sharing memory for a dimension table. The file name used to perform the open is as follows:
CTLS0369_MSG	The unix based dimension table load to memory function has been unable to open a file representing the semaphore that defines if a dimension table has been loaded for sharing memory for a dimension table. The file name used to perform the open is as follows:
CTLS0370_MSG	The unix based dimension table load to memory function has been unable to open a file representing the buffer that is used to load a dimension table into memory for sharing memory for a dimension table. The file name used to perform the open is as follows:
CTLS0371_MSG	The unix based dimension table load to memory function has been unable to memory map a file representing the mutex for sharing memory for a dimension table. The file name used to perform the memory map processing and the error number (unix errno) returned are as follows:
CTLS0372_MSG	The unix based dimension table load to memory function has been unable to memory map a file representing the semaphore that defines if a dimension table has been created for sharing memory for a dimension table. The file name used to perform the memory map processing and the error number (unix errno) returned are as follows:
CTLS0373_MSG	The unix based dimension table load to memory function has been unable to memory map a file representing the semaphore that defines if a dimension table has been loaded for sharing memory for a dimension table. The file name used to perform the memory map processing and the error number (unix errno) returned are as follows:
CTLS0374_MSG	The unix based dimension table load to memory function has been unable to memory map a file representing the buffer that is used to load a dimension table into memory for sharing memory for a dimension table. The file name used to perform the memory map processing and the error number (unix errno) returned are as follows:
CTLS0375_MSG	Function f3100_load_dim_table_to_mm_file() failed. Review Error Log for more messages.

CTLS0376_MSG	The unix based dimension table load to memory function has been unable to create a the representing the mutex for sharing memory for a dimension table. The file name used to perform the create is as follows:
CTLS0377_MSG	The unix based dimension table load to memory function has been unable to create a file representing the semaphore that defines if a dimension table has been created for sharing memory for a dimension table. The file name used to perform the create is as follows:
CTLS0378_MSG	The unix based dimension table load to memory function has been unable to create a file representing the semaphore that defines if a dimension table has been loaded for sharing memory for a dimension table. The file name used to perform the create is as follows:
CTLS0379_MSG	The unix based dimension table load to memory function has been unable to create a file representing the buffer that is used to load a dimension table into memory for sharing memory for a dimension table. The file name used to perform the create is as follows:
CTLS0380_MSG	An attempt has been made to memory map more files than allowed by the MAX_NUM_MEMORY_MAPPED_FILES setting. Please check how many memory mapped files you are trying to allocate. The name of the file that was the one that exceeded the limit is as follows:
CTLS0381_MSG	An error occurred when performing a commit. Refer to ODBC Messages in the error message log.
CTLS0382_MSG	The parameter LoadDimensionTableGroup is not set. Program CTLU012 requires that the LoadDimensionTableGroup must be set to the name of a group named on the dim_table_load_control table."
CTLS0383_MSG	Error encountered when preparing the update for the dw proc grp run log table. Refer to ODBC Messages in the error message log. The dw proc grp run log table name is as follows:
CTLS0384_MSG	Error encountered when preparing the update for the dw proc run log table. Refer to ODBC Messages in the error message log. The dw proc run log table name is as follows:
CTLS0385_MSG	Error encountered when performing the count(*) of the dw proc grp run log table. Refer to ODBC Messages in the error message log. The select statement used to count the rows in the dw proc grp run log table is as follows:
CTLS0386_MSG	Error encountered when performing an insert into the dw proc grp run log table. Refer to ODBC Messages in the error message log. The dw proc grp run log table name is as follows:
CTLS0387_MSG	Error encountered when performing an update of the dw proc grp run log table. Refer to ODBC Messages in the error message log. The dw proc grp run log table name is as follows:
CTLS0388_MSG	Error encountered when performing an update to the dw proc run log table. Refer to ODBC Messages in the error message log. The update is an effort to log the fact that the process executed successfully in the dw proc run log table. The dw proc run log table name is as follows:
CTLS0389_MSG	Error encountered when performing the validate of the dw process commands table. Refer to ODBC Messages in the error message log. The select statement used to validate the dw process commands table is as follows:
CTLS0390_MSG	Error encountered when performing the open for the dw process commands table. Refer to ODBC Messages in the error message log. The select statement used to open the dw process commands table is as follows:
CTLS0391_MSG	Error encountered when issuing the command specified for a failure of a command. Refer to the error message log. The command statement that was issues and failed is as follows:
CTLS0392_MSG	Function f3200_get_next_sequence_number failed. Review Error Log for more messages.
CTLS0393_MSG	The UseSequenceName parameter is set to Y and the SequenceName parameter is not set. If you set the UseSequenceName parameter to Y you must provide a value for the SequenceName parameter."
CTLS0394_MSG	Error encountered when performing the validate of the ctl sequence nums control table. Refer to ODBC Messages in the error message log. The select statement used to validate the ctl sequence nums control table is as follows:
CTLS0395_MSG	The preparation for the update statement for the ctl sequence nums control table failed. The fully qualified name of the the ctl sequence nums control table is as follows:
CTLS0396_MSG	The UseSequenceName parameter is set to Y and the InTable must contain a field called idw_sequence_num so that IDW can put the sequence number into this field. Please add a view column called idw_sequence_number to the InTable view. The fully qualified name of the InTable is as follows:
CTLS0397_MSG	Error encountered when performing the open for the ctl sequence nums control table. Refer to ODBC Messages in the error message log. The select statement used to open the ctl sequence nums control table is as follows:
CTLS0398_MSG	No Sequence Number row was found for the SequenceName specified in the

	SequenceName parameter. SequenceNames may be case sensitive. Please check that the SequenceName parameter and the SequenceName on the dw sequence num control table are identical. The select statement used to open the ctl sequence nums control table is as follows:
CTLS0399_MSG	Error encountered when performing the update to lock the row for parameter SequenceName for the ctl sequence nums control table. Refer to ODBC Messages in the error message log. The statement used to update the ctl sequence nums control table is as follows:
CTLS0400_MSG	Error encountered when performing the update to lock the row for parameter SequenceName for the ctl sequence nums control table. The maximum number of retries has been performed as specified by the defined variable MAX_TRIES_GET_SEQ_NUM. The default is 10 retries. Refer to ODBC Messages in the error message log. The statement used to update the ctl sequence nums control table is as follows:
CTLS0401_MSG	No locked Sequence Number row was found for the SequenceName specified in the SequenceName parameter. SequenceNames may be case sensitive. Please check that the SequenceName parameter and the SequenceName on the dw sequence num control table are identical. Please check to see if other users are updating this table manually. The select statement used to open the ctl sequence nums control table is as follows:
CTLS0402_MSG	Error encountered when performing the update to unlock the row for parameter SequenceName for the ctl sequence nums control table. Refer to ODBC Messages in the error message log. The fully qualified name of the the ctl sequence nums control table is as follows:
CTLS0403_MSG	The preparation for the select statement for OutTable failed. The fully qualified name of the OutTable is as follows:
CTLS0404_MSG	The parameter UseOrderByClause is set to Yes and the parameter OrderByClause is not set. This is invalid. If you specifiy that you wish to use an OrderByClause you must also provide the OrderByClause. Please include the OrderByClause parameter or remove the UseOrderByClause parameter.
CTLS0405_MSG	The CTLF001FileName input parameter was not set. CTLDM11 requires that this parameter is set because it is the output file from the program.
CTLS0406_MSG	The parameter SchedulerCanStartBatch is set to Yes and the parameter WaitFileName is not set. If parameter SchedulerCanStartBatch is set to Yes then WaitFileName must be set to a file that the program can open for reading and writing as well as delete.
CTLS0407_MSG	The CTLF001FileName input parameter was not set. CTLDM12 requires that this parameter is set because it is the output file from the program.
CTLS0408_MSG	The UseAlternateViewForSequenceNumberLookup parameter is set to Yes and the AlternateViewForSequenceNumberLookup is not set. If the UseAlternateViewForSequenceNumberLookup parameter is set to Yes the AlternateViewForSequenceNumberLookup parameter must be set to a view that exposes the alternate key for the target fact table.
CTLS0409_MSG	The CTLF001FileName input parameter was not set. CTLU018 requires that this parameter is set because it is the input file to the program.
CTLS0410_MSG	The CTLF002FileName input parameter was not set. CTLU018 requires that this parameter is set because it is the output file to the program.
CTLS0411_MSG	Program CTLU018 uses an output table to create the DTU self describing file format for the output file. An error was encountered when performing the validate of the output table. Refer to ODBC Messages in the error message log. The select statement used to validate the output table is as follows:
CTLS0412_MSG	Program CTLU018 uses an output table to create the DTU self describing file format for the output file. An error was encountered when performing the open of the output table. Refer to ODBC Messages in the error message log. The select statement used to open the output table is as follows:
CTLS0413_MSG	Could not open CTLF002FileName for writing. Value of CTLF002FileName is as follows:
CTLS0414_MSG	Could not prepare CTLF002FileName for writing. Value of CTLF002FileName is as follows:
CTLS0415_MSG	An error occured when loading CTLF001FileName into the document object model. Value of CTLF001FileName is as follows:
CTLS0416_MSG	No First Child Element was found in the document. CTLU018 expects the existence of a First Child Element. Please review the input file and determine the cause of the missing First Child Element. Value of CTLF001FileName is as follows:
CTLS0417_MSG	An error was encountered when writing the next record to CTLF002FileName. Value of CTLF002FileName is as follows:

CTLS0418_MSG	Program CTLU018 supports a maximum of 100 document level attributes in the decoding process. Review the input XML document and determine if there is an incorrect coding. If the document is valid and requires more than 100 document level attributes please contact support@instantbi.com and request a custom program to decode the XML document. Value of CTLF001FileName is as follows:
CTLS0419_MSG	Program CTLU018 has detected there are more attributes on an element row than there are columns in the target table being used to decode the XML document. This is considered an error. Please review the XML document and the target table to determine the correct numner of columns to create in the target table. Value of CTLF001FileName is as follows:
CTLS0420_MSG	The CTLF001FileName input parameter was not set. CTLSG01 requires that this parameter is set because it is an output file from the program.
CTLS0421_MSG	The CTLF002FileName input parameter was not set. CTLSG01 requires that this parameter is set because it is an output file from the program.
CTLS0422_MSG	The CTLF003FileName input parameter was not set. CTLSG01 requires that this parameter is set because it is an output file from the program.
CTLS0423_MSG	The CTLF004FileName input parameter was not set. CTLSG01 requires that this parameter is set because it is an output file from the program.
CTLS0424_MSG	The CTLF005FileName input parameter was not set. CTLSG01 requires that this parameter is set because it is an output file from the program.
CTLS0425_MSG	The SegmentationGeographyWhereClause parameter is not set and the SegmentationGeographyWhereClause is mandatory. Please set the parameter SegmentationGeographyWhereClause to a valid condition that will be added to the where clause for the selection of rows. Note the 'and' keyword will be added in front of this parameter.
CTLS0426_MSG	The RunSegmentationInParallel parameter is set to Yes and the LastDigit_dk_vm_customer is not set. If the RunSegmentationInParallel parameter is set to Yes the LastDigit_dk_vm_customer parameter must be set to the last digit of the dk_vm_customer value that will be selected in the parallelised batch. Please set the parameter LastDigit_dk_vm_customer to the last digit of the dk_vm_customer field that you wish to select for this particular invokation of this program.
CTLS0427_MSG	Error encountered when performing the validate of the ctl sequence nums table. Refer to ODBC Messages in the error message log. The select statement used to validate the ctl sequence nums table is as follows:
CTLS0428_MSG	Error encountered when performing the validate of the vm segmentation run log table. Refer to ODBC Messages in the error message log. The select statement used to validate the vm segmentation run log table is as follows:
CTLS0429_MSG	Error encountered when performing the validate of the vm segmentation type code table. Refer to ODBC Messages in the error message log. The select statement used to validate the vm segmentation type code table is as follows:
CTLS0430_MSG	Error encountered when performing the validate of the vm segment code table. Refer to ODBC Messages in the error message log. The select statement used to validate the vm segment code table is as follows:
CTLS0431_MSG	Error encountered when performing the open for the vm segmentation type code table. Refer to ODBC Messages in the error message log. The select statement used to open the vm segmentation type code table is as follows:
CTLS0432_MSG	Error encountered when performing the open for the vm segmentation type code table. There are no rows in this table. This is not a valid situation. The vm segmentation type code table must be populated before running the PCS Segmentation Engine. Please populate this table with the correct segmentation type codes. The select statement used to open the vm segmentation type code table is as follows:
CTLS0433_MSG	Error encountered when performing the open for the vm segmentation run log table. Refer to ODBC Messages in the error message log. The select statement used to open the vm segmentation run log table is as follows:
CTLS0434_MSG	Error encountered when performing the open for the vm segmentation run log table. There are no rows in this table. This is not a valid situation. The vm segmentation run log table must be populated before running the PCS Segmentation Engine. Please populate this table with the correct segmentation run log records. The select statement used to open the vm segmentation run log table is as follows:
CTLS0435_MSG	Error encountered when performing the open for the vm segment code table. Refer to ODBC Messages in the error message log. The select statement used to open the vm segment code table is as follows:
CTLS0436_MSG	Error encountered when performing the open for the vm segment code table. There are no

	rows in this table. This is not a valid situation. The vm segment code table must be populated before running the PCS Segmentation Engine. Please populate this table with the correct segment codes. The select statement used to open the vm segment code table is as follows:
CTLS0437_MSG	Error encountered when performing the open for the vf segment points table. Refer to ODBC Messages in the error message log. The select statement used to open the vf segment points table is as follows:
CTLS0438_MSG	Error encountered when performing the open for the vf segment recency table. Refer to ODBC Messages in the error message log. The select statement used to open the vf segment recency table is as follows:
CTLS0439_MSG	Error encountered when performing the open for the vf segment frequency table. Refer to ODBC Messages in the error message log. The select statement used to open the vf segment frequency table is as follows:
CTLS0440_MSG	Error encountered when performing the open for the vf segment monetary table. Refer to ODBC Messages in the error message log. The select statement used to open the vf segment monetary table is as follows:
CTLS0441_MSG	Error encountered when performing the open for the vf segment monetary value table. Refer to ODBC Messages in the error message log. The select statement used to open the vf segment monetary value table is as follows:
CTLS0442_MSG	Could not open CTLF001FileName for writing. Value of CTLF001FileName is as follows:
CTLS0443_MSG	Could not open CTLF002FileName for writing. Value of CTLF002FileName is as follows:
CTLS0444_MSG	Could not open CTLF003FileName for writing. Value of CTLF003FileName is as follows:
CTLS0445_MSG	Could not open CTLF004FileName for writing. Value of CTLF004FileName is as follows:
CTLS0446_MSG	Could not open CTLF005FileName for writing. Value of CTLF005FileName is as follows:
CTLS0447_MSG	Could not prepare CTLF001FileName for writing. Value of CTLF001FileName is as follows:
CTLS0448_MSG	Could not prepare CTLF002FileName for writing. Value of CTLF002FileName is as follows:
CTLS0449_MSG	Could not prepare CTLF003FileName for writing. Value of CTLF003FileName is as follows:
CTLS0450_MSG	Could not prepare CTLF004FileName for writing. Value of CTLF004FileName is as follows:
CTLS0451_MSG	Could not prepare CTLF005FileName for writing. Value of CTLF005FileName is as follows:
CTLS0452_MSG	Error encountered when performing the open for the vw product points unload table. Refer to ODBC Messages in the error message log. The select statement used to open the vw product points unload table is as follows:
CTLS0453_MSG	Error encountered when performing the open for the vw product recency unload table. Refer to ODBC Messages in the error message log. The select statement used to open the vw product recency unload table is as follows:
CTLS0454_MSG	Error encountered when performing the open for the vw product frequency unload table. Refer to ODBC Messages in the error message log. The select statement used to open the vw product frequency unload table is as follows:
CTLS0455_MSG	Error encountered when performing the open for the vw product monetary unload table. Refer to ODBC Messages in the error message log. The select statement used to open the vw product monetary unload table is as follows:
CTLS0456_MSG	Error encountered when performing the open for the vw product monetary value unload table. Refer to ODBC Messages in the error message log. The select statement used to open the vw product monetary value unload table is as follows:
CTLS0457_MSG	Error encountered when performing the open for the vw product points unload table. There are no rows in this table. This is not a valid situation. The select statement used to open the vw product points unload table is as follows:
CTLS0458_MSG	Error encountered when performing the open for the vw product recency unload table. There are no rows in this table. This is not a valid situation. The select statement used to open the vw product recency unload table is as follows:"
CTLS0459_MSG	Error encountered when performing the open for the vw product frequency unload table. There are no rows in this table. This is not a valid situation. The select statement used to open the vw product frequency unload table is as follows:
CTLS0460_MSG	Error encountered when performing the open for the vw product monetary unload table. There are no rows in this table. This is not a valid situation. The select statement used to open the vw product monetary unload table is as follows:
CTLS0461_MSG	Error encountered when performing the open for the vw product monetary value unload table. There are no rows in this table. This is not a valid situation. The select statement used to open the vw product monetary value unload table is as follows:
CTLS0462_MSG	Error encountered when writing CTLF001FileName. Value of CTLF001FileName is as follows:
CTLS0463_MSG	Error encountered when writing CTLF002FileName. Value of CTLF002FileName is as

	follows:
CTLS0464_MSG	Error encountered when writing CTLF003FileName. Value of CTLF003FileName is as follows:
CTLS0465_MSG	Error encountered when writing CTLF004FileName. Value of CTLF004FileName is as follows:
CTLS0466_MSG	Error encountered when writing CTLF005FileName. Value of CTLF005FileName is as follows:
CTLS0467_MSG	Error encountered when performing the open for the vf segment rows table. Refer to ODBC Messages in the error message log. The select statement used to open the vf segment rows table is as follows:
CTLS0468_MSG	Error encountered when performing the open for the vw all pcs segments unload table. Refer to ODBC Messages in the error message log. The select statement used to open the vw all pcs segments unload table is as follows:
CTLS0469_MSG	Error encountered when performing the open for the vw all pcs segments unload table. There are no rows in this table. This is not a valid situation. The select statement used to open the vw all pcs segments unload table is as follows:
CTLS0470_MSG	Function f2100_process_pcs_segments failed. Review Error Log for more messages.
CTLS0471_MSG	Function f2100_process_segment_points failed. Review Error Log for more messages.
CTLS0472_MSG	Function f2110_process_segment_recency failed. Review Error Log for more messages.
CTLS0473_MSG	Function f2120_process_segment_frequency failed. Review Error Log for more messages.
CTLS0474_MSG	Function f2130_process_segment_monetary failed. Review Error Log for more messages.
CTLS0475_MSG	Function f2140_process_segment_monetary_value failed. Review Error Log for more messages.
CTLS0476_MSG	The CTLF001FileName input parameter was not set. CTLSG02 requires that this parameter is set because it is the output file for the program.
CTLS0477_MSG	An error occured when processing the following sql statement directly against the database. You may need to turn on ODBC tracing to detect the nature of the error. The sql statement that was issued is as follows. Note that the sql statement recorded here might be truncated to 4,000 characters:
CTLS0478_MSG	The WriteDeleteRecordsToFile parameter is set to Yes and the CTLF001FileName parameter is not set. If you set the WriteDeleteRecordsToFile parameter to Yes you must provide a value for the CTLF001FileName parameter so that the delete records can be written to the file.
CTLS0479_MSG	The CTLF001FileName input parameter was not set. CTLU019 requires that this parameter is set because it is the input file to the program.
CTLS0480_MSG	The file CTLF001FileName is empty. Program CTLU019 should not be executed if there has been no data input to CTLU019. Value of CTLF001FileName is as follows:
CTLS0481_MSG	Error encountered when performing the delete of a record found in CTLF001 from OutTable. Refer to ODBC Messages in the error message log. The fully qualified table name for the OutTable table is as follows:
CTLS0482_MSG	The movement of data from the file CTLF001 to OutTable failed. The fully qualified name of the OutTable is as follows:
CTLS0483_MSG	Error encountered when performing the open for the vf segment stats table. Refer to ODBC Messages in the error message log. The select statement used to open the vf segment stats table is as follows:
CTLS0484_MSG	Error encountered when determining the number of result columns for file CTLF001. Value of CTLF001FileName is as follows:
CTLS0485_MSG	Error encountered when performing the open for the vw segment points stats 03 table. Refer to ODBC Messages in the error message log. The select statement used to open the vw segment points stats 03 table is as follows:
CTLS0486_MSG	Error encountered when performing the open for the vw segment points stats 03 table. There are no rows in this table. This is not a valid situation. The vw segment points stats 03 table must be populated by running the PCS Segmentation Engine. The select statement used to open vw segment points stats 03 table is as follows:
CTLS0487_MSG	Error encountered when performing the validate of the vm stats type code table. Refer to ODBC Messages in the error message log. The select statement used to validate the vm stats type code table is as follows:
CTLS0488_MSG	Error encountered when performing the open for the vm stats type code table. Refer to ODBC Messages in the error message log. The select statement used to open the vm stats type code table is as follows:
CTLS0489_MSG	Error encountered when performing the open for the vm stats type code table. There are no

	rows in this table. This is not a valid situation. The vm stats type code table must be populated before running the PCS Segmentation Engine. Please populate this table with the correct segment codes. The select statement used to open the vm stats type code table is as follows:
CTLS0490_MSG	The parameter AllowDataConversionErrors is set to yes and the parameter defining the ErrorFileName is not set. When the parameter AllowDataConversionErrors is set to Yes the ErrorFileName parameter must be set to a file that the program can open for writing.
CTLS0491_MSG	Could not write to ErrorFileName. Value of ErrorFileName is as follows:
CTLS0492_MSG	Function f2100_load_lif_file_to_table failed. Review Error Log for more messages.
CTLS0493_MSG	The CTLF001FileName input parameter was not set. CTLU020 requires that this parameter is set because it is the input file to the program.
CTLS0494_MSG	Function f1000_initialise_part3 failed. Review Error Log for more messages.
CTLS0495_MSG	The parameter ProcessCommand is not set. The ProcessCommand must be set to a Process Command that is defined in the ctl pocess commands table.
CTLS0496_MSG	The function f3000_process_command_parameter issued an error return code. Review the error message log for more information as to the cause of this error code. The command to be executed is as follows:
CTLS0497_MSG	Error encountered when performing the prepared select. Refer to ODBC Messages in the error message log. The fully qualified table name for the select table is as follows:
CTLS0498_MSG	Function f2000_fetch_sql_statements failed. Review Error Log for more messages.
CTLS0499_MSG	Error encountered when performing the validate of the ctl dw sql stmts table. Refer to ODBC Messages in the error message log. The select statement used to validate the ctl dw sql stmts table is as follows:
CTLS0500_MSG	Error encountered when performing the validate of the ctl dw sql stmt log table. Refer to ODBC Messages in the error message log. The select statement used to validate the ctl dw sql stmt log table is as follows:
CTLS0501_MSG	Error encountered when performing the open of the ctl dw sql stmts table. Refer to ODBC Messages in the error message log. The select statement used to open the ctl dw sql stmts table is as follows:
CTLS0502_MSG	The ctl dw sql stmts table has been found to have more than one SQL Statement that matches the passed SQL FileName Parameter. This is not allowed. The select statement used to read the ctl dw sql stmts table is as follows:
CTLS0503_MSG	Error encountered when performing the read of the ctl dw sql stmts table. The select statement used to read the ctl dw sql stmts table is as follows:
CTLS0504_MSG	Error encountered when performing the reading of the number of columns for the ctl dw sql stmts table. The select statement used to read the ctl dw sql stmts table is as follows:
CTLS0505_MSG	Error encountered when performing an SQL Statement in CTLU023. Refer to ODBC Messages in the error message log. The SQL statement issued is as follows:
CTLS0506_MSG	Error encountered when performing the validate of the ctl proc grp start run log table. Refer to ODBC Messages in the error message log. The select statement used to validate the ctl proc grp start run log table is as follows:
CTLS0507_MSG	Error encountered when performing the prepare for the insert into the ctl proc grp start run log table. Refer to ODBC Messages in the error message log. The fully qualified name of the table used to prepare the ctl proc grp start run log table for inserting is as follows:
CTLS0508_MSG	Error encountered when performing the insert into the ctl proc grp start run log table. Refer to ODBC Messages in the error message log. The ctl proc grp start run log table name is as follows:
CTLS0800_MSG	Metadata mismatch detected. The source field is longer than the target field. This may cause abnormal termination of the ETL processing. The source and target fields are as follows:
CTLS0801_MSG	Metadata mismatch detected. The source field is nullable and the target field is not nullable. This may cause abnormal termination of the ETL processing. The source and target fields are as follows:
CTLS0802_MSG	Metadata mismatch detected. The source field is not a DATE field and the target field is a DATE field. This may cause abnormal termination of the ETL processing. The source and target fields are as follows:
CTLS0803_MSG	Metadata mismatch detected. The source field is not a TIMESTAMP field and the target field is a TIMESTAMP field. This may cause abnormal termination of the ETL processing. The

	source and target fields are as follows:
CTLS0804_MSG	Metadata mismatch detected. The source field is some kind of character or binary field and the target field is some kind of numeric, float or integer field. This may cause abnormal termination of the ETL processing. The source and target fields are as follows:
CTLS9001_MSG	An ODBC Error Occurred. Details Following.

2.1. Notes on Error Messages

SeETL[RT] performs many checks during processing to determine if there are any error conditions that have occurred. All error conditions detected are reported via the error message classes. When running any program you can specify where the error messages are written to:

1. The C error file which is the console.
2. A file accessible by the server. All messages are always appended to the file if the file already exists.
3. To the error messages table in the data warehouse.

It is recommended that all error messages are written to the error message table in the data warehouse in a production environment because this is the easiest place for support staff to find error messages.

However, during initial setup and testing one of the errors that you may have is the inability to connect to the data warehouse database. In which case it will not be possible to write an error message to the database!!

The ODBC database access classes do not issue any error messages themselves. Any error conditions raised by ODBC are sent back to the calling program so that the calling program can add information to the error condition (such as function in which the error was found) prior to writing it to the error output specified.

This introduces a complication for those functions in the ODBC classes that emulate database functions by loading the dimension tables into memory and then searching the in memory arrays. There are a variety of problems that can arise during the execution of this code, the most notable of which is running out of memory.

In any case where the ODBC database access classes detect an error which is not an ODBC error the class issues an error message that is 'reported as' an ODBC error. You can identify these messages because the SQL State and SQL Native Error values are set to 9999, which you will not find in there ODBC error message manual. Currently these are messages CTLS0116 to CTLS0124. However, more messages which are reported as ODBC errors are likely to be added in the future.

2.2. The Unix Errno Code

The unix memory mapped io code returned the unix errno error code in the event of some form of invalid or failed call to perform memory mapping. These codes may change slightly in meaning from one version of unix to another. SeETL[RT] simply prints the number of the code. Should you receive an error message quoting the errno from your unix operation system we recommend you consult your operating system documentation to determine the meaning of the errno for your operating system.

General values of errno are as follows:

```
#define EPERM           1
#define ENOENT          2
#define ESRCH           3
#define EINTR           4
#define EIO             5
#define ENXIO           6
#define E2BIG           7
#define ENOEXEC         8
#define EBADF           9
#define ECHILD          10
#define EAGAIN          11
#define ENOMEM          12
#define EACCES          13
#define EFAULT          14
#define EBUSY           16
#define EEXIST          17
#define EXDEV           18
#define ENODEV          19
#define ENOTDIR         20
#define EISDIR          21
#define EINVAL          22
#define ENFILE          23
#define EMFILE          24
#define ENOTTY          25
#define EFBIG           27
#define ENOSPC          28
#define ESPIPE          29
#define EROFS           30
#define EMLINK          31
#define EPIPE           32
#define EDOM            33
#define ERANGE          34
#define EDEADLK         36
#define ENAMETOOLONG    38
#define ENOLCK          39
#define ENOSYS          40
#define ENOTEMPTY       41
#define EILSEQ          42
```

3. APPENDIX 2 - METADATA REPORTS

The following tables list the metadata reports currently available.

Run Time Reports	
Report Name	**Description**
MTR0101A Recent Audit Records	This report lists recent audit records from the Data Warehouse. You can use this report to see how many rows were processed by programs recently.
MTR0101B Recent Error Messages	This report lists recent error messages from the Data Warehouse. You can use this report to see the different types of error messages that have been produced recently.
MTR0101C Batch Control	This report lists the batch control table. This shows the date on which the batch ran, the batch number and also lists the finished status of the batch.
MTR0101D Batch Pre-Requisites	This report lists the pre-requisites for batches. When the pre-requisites for the batch are met for the current batch the batch will be started by the scheduler.
MTR0101E Process Group Pre-Requisites	This reports lists the pre-requsites for process groups. When the pre-requisites for the process group are met the process group will be started by the scheduler.
MTR0101F Batch Schedule	This report lists the commands that are run in a batch. It is drillable by batch and process group to list the individual commands for each process group.
MTR0101G Batch Schedule Combined	This report combines the lists of the batch and process groups pre-requisites and the commands to be run in the batch and process group. It is drillable by batch and process group. It is the combination of reports MTR0101D, MTR0101E, MTR0101F.
MTR0101H Batch Run Log Combined	This report lists run log for a single batch which is selected using a parameter. It lists the batch run log, the process group run log and the command run log for the single batch.
MTR0101I Batch Run Log By Status.	This report lists run log for a single batch which is selected using a parameter. It lists the batch run log, the process group run log and the command run log for the single batch.
MTR0101J – Executed SQL Statements	This report lists the SQL Statements that have been executed in the selected batches. It lists them in the timestamp order in which they were executed. It reports the start timestamp, stop timestamp, elapsed time, rows processed and calculates rows processed per second. It is a very handy report to track the execution of SQL in the ETL subsystem.

ETL Design Reports	
Report Name	**Description**
MTR0102A Aggregate Keys Control	This report lists the last key used by the SeETL RT for all the dimension and association table.
MTR0102B Aggregation Control	This report lists the levels at which dimension tables will be aggregated for specific dimension tables. Each dimension table may be at a summary level of 0-9. The maximum number of dimension tables that can be linked to a single fact table is 50.
MTR0102C Dimension Table Key Definitions.	This report lists all the dimension tables, the type of dimension table it is (Type 1 or Type 2) and then lists the columns that are used from the source table to construct the real key that is used by SeETL RT as the alternate primary key for the dimension table. The report also lists how many fields are used to construct the real key.
MTR0102D Type 2 Dimension Table Column Definitions	This report lists all the Type 2 dimension tables. It lists the table name and the columns on the input table to test for change as part of the Type 2 dimension table processing.
MTR0102E Dimension Table Load Control	This report lists the load control options for dimension tables. Because dimension tables can be loaded into memory based on fact/dimension table combinations if so desired the report lists the fact table name, dimension table name and load control option. If the dimension table is to be loaded into memory mapped IO the load control group defines which groups of dimension tables will be loaded together.
MTR0102F ODBC Data Types	The SeETL RT prints the ODBC data type in the header of the self-describing file format. This table lists the integer used by ODBC as the data type and a short description of the data type. This assists the user to easily refer to the data types of the fields in working files.
MTR0102G Month Control	This report lists the one row from the Month Control table. This lists the first day of the month for the last, current and next months. This is to allow the user to set the first day of the month to a different date than the first of the calendar month. The report also lists the high and low value dates which are used as default dates when no date is provided.

Miscellaneous Reports	
Report Name	**Description**
MTR0103A - Detailed Commands	This report lists the detailed commands that may be run by the SeETL RT Scheduler. This table is used to 'hide' the commands being processed against the data warehouse and to remove the need the place commands into files in the file system. The commands in this table co-relate to the commands in the Commands Table listed in report MTR0101C - Batch Schedules.
MTR0201A - DataStage Parameters	This report lists the DataStage parameters that can be set and passed to the DataStage Job Submission Utility - With Parameters (CTLU010). This utility can be started by the SeETL RT Scheduler and the parameters from this table can alter the processing performed by those jobs. The main use of the table is to allow the migration of jobs between environments with no change to the DataStage jobs.

Metadata Reports	
Report Name	**Description**
MDA0101A - Source to Target Mappings By Staging	This report lists the source to target mappings from the mapping spreadsheet sorted by the row number of the mapping row in the mapping spreadsheet. It is fully drillable and mappings can be selected depending on many parameters.
MDA0101B - Target To Source Mappings By Presentation Views	This report lists the source to target mappings from the mapping spreadsheet sorted by the row number of the mapping row but with the presentation view information as the leading columns. This report allows the user to see what data goes into the presentation view. It is fully drillable and mappings can be selected depending on many parameters.
MDA0101C - Target To Source Mappings By Presentation Views	This report lists the source to target mappings from the mapping spreadsheet sorted by the Presentation View/Column number of the mapping row. This report allows the user to select and see what data goes into the presentation view driven by the presentation view name itself. It is fully drillable and mappings can be selected depending on many parameters.
MDA0101D - Source Data Descriptions	This report lists the descriptions of columns and tables for the source data. The descriptions have not been produced on other reports because they may take up a significant amount of space. The report is drillable by variables in the staging area.
MDM0101A - Data Model Number of Languages	This report lists the number of languages that the data models have been developed in for the current client. IBI Supplies only English as a base language.
MDM0101B - Data Model Target Directories	This report lists the target directories to which sql generation language will be written to create views.
MDM0101C - View Descriptions	This report lists the view descriptions of the views in the BI4ALL Data Models.
MDM0101D - Column Descriptions	This report prints the details of views and columns in the BI4ALL Data Models.
MDM0101E - Column Descriptions	This report prints the details of views and columns in the BI4ALL Data Models.
MDM0101F - Data Model Joins	This report prints the details of joins between tables in the BI4ALL Data Models.
MDM0101G - Other Views	In BI4ALL there are sometimes views that need to be created that are not part of the data model. They are views for ETL processing that simplify the ETL processing defined in the workbook. These views became known by the name of "other views". There is a sheet for them but it is rarely used. It is starting to become more widely used and so we are putting this report over the top of the other views table.
MDM0101H - Compound Views	In BI4ALL there are sometimes views that need to be created that are not part of the data model. They are views for ETL processing that simplify the ETL processing defined in the workbook. In rare cases these views must be compound views using such things as Union clauses. There is a sheet for them but it is rarely used. It is starting to become more widely used and so we are putting this report over the top of the compound views table.
MDM0101I - Table Definitions	This report documents the tables that are defined in the create tables worksheet for which Create Table DDL will be generated by SeETL ^{DT}.
MDP0101A - Row Counts	This report is a Data Profiling Report that lists the number of rows in a table over a period of time.
MDP0101B - Column Analysis	This report is a Data Profiling Report that lists the profiling analysis for columns.
MDP0101C- Column Cross Reference Analysis	This report is a Data Profiling Report that lists the cross reference analysis for columns.
MIN0101A - Infa Mappings Header Documentation	This report prints Informatica Mapping Documentation to be provided to programmers for writing Informatica mappings. It provides just heading level information and does not provide detailed mapping information. It should be used to print documentation where there is no need to document detailed movements of data. It is typically used for source to staging mappings where little change takes place in the actual mappings.
MIN0102A - Infa Mappings Detailed Documentation	This report prints Informatica Mapping Documentation to be provided to programmers for writing Informatica mappings. It provides the detailed mappings at a field level. It should be used to print documentation where there is a need to document detailed movements of data. It is typically used for staging to ODS or

	EDW mappings.
MIN0103A - Infa Mappings Audit Mappings Documentation	This report prints Informatica Mapping Documentation to be provided to analysts to determine the fields to be audited.
MPR0101A - Project Plan Combined	This report lists the Project Plan pretty much as it is entered via the spreadsheet. It is simply a report that will allow the spreadsheet data loaded into the dictionary to be downloaded again to another spreadsheet in a more readable format for the user.
MRP0101A - Report Descriptions	This report lists the reports that are documented.
MRP0101B - Report IO Descriptions	This report prints the details of the inputs and outputs to reports. It is meant as a tool that will assist end users find the reports that they are looking for.
MSP0101A - Stored Procedure Translations Combined	This report lists the Stored Procedure Translation Directives. The SeETL[DT] Tool can now translate stored procedures that contain the source code for many language derivatives and it can send those language derivatives to various output files based on the setting in the spreadsheet from which this report is derived. This report is simply to make visible the stored procedure translations that are possible via the tool.
MSP0101B - Stored Procedure Descriptions	This report lists the view descriptions of the views in the BI4ALL Data Models.
MSP0101C - Stored Procedure IO Descriptions	This report prints the details of Inputs and Outputs of Stored Procedures. It is a report to be used to find and use stored procedures that are known to be certified and working so that they can be linked to new front end tools and reports.
MSQ0101A - Count Input Rows SQL	This report documents the input views that will have count SQL generated for them by the SeETL[DT] software. This count SQL is an integral part of the audit processing that is recommended in IBI designed data warehouses.
MSQ0101B - Run Statistics	This report documents the input to create run statistics commands against the database. The commands are as they will be executed against the database. The commands can be loaded into the database and executed as part of a schedule that is executed by the scheduler. This allows the recurring creating of statistics to be made a part of the overall schedule for the data warehouse.
MSQ0101C - Create Table As Commands	This report documents the input to run create table as commands against the database. The commands are as they will be executed against the database. The commands can be loaded into the database and executed as part of a schedule that is executed by the scheduler. This allows the recurring creating of tables with create table as to be made a part of the overall schedule for the data warehouse.
MSQ0101D - Create Index	This report documents the input to run create index commands against the database. The commands are as they will be executed against the database. The commands can be loaded into the database and executed as part of a schedule that is executed by the scheduler. This allows the recurring creating of indexes with the create index to be made a part of the overall schedule for the data warehouse.
MSQ0101E - Create Storage Objects	This report documents the input to create storage objects commands against the database. The commands are as they will be executed against the database. The commands can be loaded into the database and executed as part of a schedule that is executed by the scheduler. This allows the recurring creation of storage objects to be made a part of the overall schedule for the data warehouse.
MSQ0101F - Update Permissions	This report documents the input to run update permissions commands against the database. The commands are as they will be executed against the database. The commands can be loaded into the database and executed as part of a schedule that is executed by the scheduler. This allows the recurring updating of permissions on objects to be made a part of the overall schedule for the data warehouse.
MSQ0101G - Create Generic Objects	This report documents the input to create generic objects commands against the database. The commands are as they will be executed against the database. The commands can be loaded into the database and executed as part of a schedule that is executed by the scheduler. This allows the recurring creation of generic objects to be made a part of the overall schedule for the data warehouse.
MSQ0101H - SQL Statements	In SeETL[DT] it is not possible to load SQL Statements into the database and have them executed from inside the database. These statements are then listed in this report.
MTR0102E - Source File Definitions	This report lists all the Input Files that have been taken into consideration for inclusion into the data warehouse. They are listed here even if they have been rejected for inclusion to complete the documentation for all tables and files considered.

3.1. Screen Shots of MetaData Reports

Removed as sampled reports are available from the Instant Business Intelligence web site..

http://www.instantbi.com/LinkClick.aspx?link=SeETL+Report+Descriptions.pdf&tabid=59&mid=448
http://www.instantbi.com/LinkClick.aspx?link=SeETL+Example+Reports.zip&tabid=59&mid=448

4. APPENDIX 3 – LINES OF CODE OF SEETL

SeETL[RT] is a tool that has evolved over many years culminating in an easy to use suite of programs that can be used to build quite sophisticated data warehouses.

We feel that a very good way of demonstrating the depth of the code delivered with SeETL[RT] is to publish the 'lines of code' count.

Please note that 'lines of code' in this instance are the lines of text in the file including comments and blank lines for formatting.

Program Name	Program Description	Lines of Code
Source Code Counts for SeETL Run Time in C++		
CTLDM01	This is the type 1 dimension table processing program.	2,425
CTLDM11	This is the type 1 dimension table processing program for initial load.	1,921
CTLDM02	This is the type 2 dimension table processing program.	3,489
CTLDM12	This is the type 2 dimension table processing program for initial load.	2,534
CTLAT01	This is the attribution program.	2,243
CTLAT02	This is the attribution program for Sybase IWS.	2,614
CTLAG01	This is the aggregation program.	2,060
CTLCL01	This is the consolidation program.	1,173
CTLAS01	The is the dimension table association processing program.	2,751
CTLPR01	The is a Sybase IWS specific program.	2,755
CTLBM01	The Batch Maintenance Utility.	601
CTLBM02	The Batch Maintenance Utility.	428
CTLU001	The Data Transfer Utility. A utility to move data between ODBC data sources.	4,656
CTLU002	The SQL Statement Processor Utility.	433
CTLU003	The SQL Server 2000 DDL Generator Utility	1,093
CTLU004	The Oracle 8/9 DDL Generator Utility.	1,093
CTLU005	The Delimiter Separated Values Reformat Utility.	918
CTLU006	The Fixed File Format Reformat Utility.	985
CTLU007	The Generate Delta File Utility.	1,687
CTLU008	The Batch Processing Scheduling Utility.	5,717
CTLU009	The Process Group Manager Utility.	3,089
CTLU010	The DataStage Job Submission Utility.	761
CTLU011	The DataStage Job Submission Utility – With Parameters	1,261
CTLU012	Load Dimension Tables into Memory Maps Utility.	2,622
CTLU013	The MetaData Checking Utility.	747
CTLU014	The MetaData Printing Utility.	485
CTLU015	The MetaData Loading Utility.	680
CTLU016	The Data Correction Utility.	1,995
CTLU017	Batch Processing Sleep Utility	210
CTLU018	XML File Reformat Utility.	945
CTLU019	Delete Target Rows Utility.	769
CTLU020	Bulk Load Rows into SQL Server Utility.	721
CTLU021	Execute ProcessName From Scheduler Utility.	843
CTLU022	The Batch Processing Scheduling Utility – Increased Resilience	5,750
CTLU023	Process SQL Statements loaded into the Data Warehouse	1,322
CTLSG01	Segmentation – The Segment Data Creation Program	4,219
CTLSG02	Segmentation – The Pivot Program.	2,045
CTLSG03	Segmentation – The Statistics Generation Program.	943
CTLSG04	Segmentation – The Statistics Calculation Program	1,656
Total C++		**72,639**

Source Code Counts for SeETL Design Time in vb.net		
SeETL	The VB.NET code for SeETL[DT].	70,203
LicenseMgt	The VB.NET code for License Management.	1,525
Parser	The VB.NET code for Parsing in SeETL Design Time..	553
Console	The VB.NET code for Console interface in SeETL Design Time..	107
Voucher	The VB.NET code for Voucher Management in SeETL Design Time..	1,171
Total VB.net		**73,559**
Total SeETL		**146,198**

The major classes and their names are listed below.

File Name	Class Description	Lines of Code
CTLAudit	This is the Audit Class. It provides the Auditing Services for the SeETL [RT].	299
CTLMessages	This is the Message Class. It provides Messaging Services for the SeETL [RT].	566
CTLFileAccess	This is the File Accessing Class. It provides File Access Services for the SeETL [RT].	2,633
CTLODBCAccess	This is the ODBC Accessing Class. It provides all ODBC Access Services for the SeETL [RT]. CODBCAccess is the 'mother of all classes'.	6,497
Dwdwdefs.h	This is the definitions include file. It sets all #defines for the SeETL [RT].	824
Dwdwparm.h	This is the parameter processing include file. It provides support for all parameter definitions and decoding for all SeETL [RT] programs.	2,115
Dwdwfunctions.h	This is the C functions include file. It contains all non class specific functions required to support the SeETL [RT].	2,075
Dwdwfargv.h	This is the argv decoding file for SeETL [RT]. It extracts parameters from argv.	73
Dwdwcopy.h	This is the header file that includes all the other header files for SeETL [RT].	135
Dwdwvars1.h	This is the header file that includes all the variables that are global for SeETL [RT].	448
Dwerrinit1.h	This is the header file that includes initialisation for messages for SeETL [RT].	136
Total		**15,801**

Total lines of Code at 1.5.5. release = 39,181.
Total lines of Code at 2.0. release = 48,461.
Total lines of Code at 2.1 beta a = 57,222.
Total lines of Code at 2.1 Release = 61,031.
Total lines of Code at 3.0.00 Release = 73,993.
Total lines of Code at 3.1.00 Release = 125,347.
Total lines of Code at 3.1.01 Release = 142,863.
Total lines of Code at 3.1.02 Release = 161,999.

The total number of lines of code currently stands at ***161,999!!!***

And not only is this a lot of code, it is the 'right code' to build your data warehouse!! It embeds a very large amount of experience gained over many years into a single suite of source code with which to build your data warehouse.

If you chose the 'Source Code' License Agreement this code will stand you in good stead for all your data warehouse data movement requirements.

5. APPENDIX 4 – ENHANCEMENTS FOR 1.5.5

SeETL[RT] Version 1.5.5 contains the following enhancements:

Using Quote Characters

The parameter UseQuoteCharacter was created in a previous upgrade. However, the method was not set in all the programs that access files. In this release all the files access are passed the UseQuoteCharacter parameter.

Using a Substitute Character to Represent Null Characters

And today I had cause to try to create a Load Interface File in DTU and remove the reference of the Null Character.

In 1.5.4. the UseNullCharacter=No parameter did not remove the null character. It reverted back to the default null character instead. The CFILERecordset Class method `LIF_WriteRow` had to be updated to completely ignore the null character when writing the load interface file record.

For those of you with the source code, the code fragment in `LIF_WriteRow` now looks like this:

```
IF (ptr_table_field_desc_array[index1].field_is_null EQUALS_ TRUE) THEN
  BEGIN_
    IF (ws_UseNullCharacter EQUALS_ TRUE) THEN
        BEGIN_
            strcat(ws_io_buffer,ws_NullCharacter) ;
        END_
  END_
ELSE
  BEGIN_
      strcat(ws_io_buffer,ws_field_data) ;
  END_
```

Note that if the field is null then nothing is written to the ws_io_buffer and so nothing is written to the output file. So the file will contain two delimiters next to each other. If the UseQuoteCharacter=Yes parameter is specified then there will be two of the quote characters together with no extra character between them.

This is what the database loader is expecting…

InputTableOrFileCanBeEmpty Parameter

In 1.5.4. this parameter could be set to ‘Y’ or ‘N’. This was failing to maintain the standard that all Y/N fields are set to ‘Yes’ or ‘No’ by the user and interpreted to TRUE or FALSE inside the code. The valid values of this parameter are now ‘Yes’ or ‘No’.

Support of Real Data Types by the Consolidation Program CTLCL01

When I first wrote the consolidation program the 'real' data type was not supported as one of the data types that could be consolidated on a table. The reason for this was pretty simple. When a 'real' data type is returned from ODBC it does not tell you how many decimal places can be in the number so it was a little difficult to put the consolidation code into the program. I would have to 'default' the number of decimal places which is not really what the customer would like (I thought).

Well, a customer asked if he could use real data types in the consolidation process. The answer is 'Yes, and you need to default the number of decimal places used. He chose 10 decimal places. So, now CTLCL01 supports real data types with 10 decimal places hard coded inside the program.

The code fragment looks like this:

```
IF (ptr_CTLF002->ptr_table_field_desc_array[index1].field_type EQUALS_ SQL_REAL) THEN
  BEGIN_
      ws_field_decimal_digits = 10 ;
  END_
ELSE
  BEGIN_
      ws_field_decimal_digits = ptr_summary_fact_table->
                          ptr_table_field_desc_array[index1].field_decimal_digits ;
  END_
```

If the field is a double or a decimal then the number of decimal places specified in the field definition will be used. If the field is real, then 10 decimal places will be used.

Case Sensitivity in Dimension Table Key Lookups

Case sensitivity is always a problem with dimension table lookups. The simple fact of the matter is that Oracle is case sensitive when you perform where clauses and there's not much to be done about that (2004). If you are using Oracle you must make sure that you set the case of the dim_char_ky_fld and the dimension_table_char_key in the incoming fact table view to be the same if you plan to perform the lookup using Oracle against the database.

If you want a quick and easy way to get around this, in 1.5.5. I am introducing case insensitivity in the 'in-memory' lookups no matter what database is used. I have also made sure any trailing blanks are removed. So, if you have the memory and want to avoid the whole case sensitive and trailing blanks problem you can set the ctl_dim_table_load_control options for the specific dimension tables and fact tables you would like to load into memory for lookups.

I know this will be helpful as it's been an issue with a number of ETL tools that I have used.

Ability to Default Values When Loading Them with the DTU

Often times when transferring data from a source system to a target staging area I take the source data and add some specific columns to the staging area tables such as ROW_DELETED_FROM_SRC and ROW_VALID_IND. These fields need to be defaulted. With the DTU I usually create the fields as not nullable with a default and then put a view over the target table to hide the fields I want to default from the DTU so that they will default properly.

However, I thought it was about time I introduced the ability to specifically default fields on the output table for the DTU. This way there is no need to create a view over the top of the tables in the staging area. Also, it provides the ability to set the fields to default values for the Load Interface File options of the DTU.

I've decided to make it possible to default 10 fields. If someone wants more please write to us at support@instantbi.com. The parameters that have been added are:

- SetDefaultField1-10=Yes or No
- SetDefaultFieldName1-10=<field name>
- SetDefaultFieldValue1-10=<field value>

The field value should not contain blanks. Blanks in the field value are not supported.

Allowing Files to be Processed to be Empty

The earlier versions of the SeETL [RT] and utilities did not allow files to be empty. In 1.5.4. an upgrade was put in place to allow the Data Transfer Utility to successfully process empty files. However, with the advent of the scheduler (rather than scripting) it has become more necessary to allow all SeETL [RT] programs to successfully process empty files and return a zero return code. This is because the scheduler only allows the continual processing of processes that return zero return codes. There is no concept of various return codes being valid.

Thus, in 1.5.5 it is possible to specify InputTableOrFileCanBeEmpty=Yes for all SeETL [RT] programs. The behaviour you should expect is that if CTLAT01 is passed an empty view it will produce a CTLF001 file that is empty. That is, there are zero bytes in the file. Importantly there is NO HEADER RECORD in the CTLF001 file. CTLAG01, CTLCL01 and CTLU001 will then each recognise the empty CTLF001 file and produce/read empty files as appropriate as long as the parameter InputTableOrFileCanBeEmpty=Yes is specified.

The SeETL [RT] programs can be scheduled using the scheduler and when InputTableOrFileCanBeEmpty=Yes is specified they will process empty files successfully. That is, they will run and return zero return codes.

I am still of the opinion that processing empty files is something of a worry so each program will produce a warning message to warn the user that an empty file has been processed and the user has specified the parameter InputTableOrFileCanBeEmpty=Yes. If the user does not want to successfully process empty files then the user should omit the InputTableOrFileCanBeEmpty parameter and it will default to No. This will cause an error to be signalled and the processing will stop with a severe error message issued to the error message log.

6. APPENDIX 5 – ENHANCEMENTS FOR 1.6.1

6.1. Performance Related Enhancements

The point release for SeETL[RT] Version 1.6.1 is all about two things:

Scalability
and
Performance

SeETL[RT] came out of my own development work. Early versions where in cobol and these were implemented in production environments for very large customers. As windows PC have sped up it became clear that quite sizable data warehouses could be supported using a tool like SeETL[RT] on windows.

When it was re-designed and converted from Cobol to C++/ODBC SeETL[RT] was designed for **maximum developer productivity and data model flexibility.** It was **not** designed for **maximum scalability**.

It was assumed that the speeding up of processors and the availability of cheap memory would mean that SeETL[RT] would have a comfortable niche in the size of the Data Warehouses that it was used to implement. And it has. I also assumed that clients that were 'too big' for SeETL[RT] would buy one of the vendor ETL tools, and they have.

The **surprise** has been that SeETL[RT] was used on some quite large volumes for prototyping and a number of prospective customers are looking at volumes that are well above the original limits of SeETL[RT]. Because SeETL[RT] can be licensed as source code is 'almost free' when compared to the vendor ETL tools for a large company. And 'almost free' is a compelling price for a large company.

We have even used SeETL[RT] to prototype the full volume financial data warehouse for a large telecommunications company. This version of SeETL[RT] was originally intended to be able to load around 3M transactions per night. It was not thought that it would be used to build very large Data Warehouses.

Given this was happening it was decided to update and add features which would improve the performance and scalability of SeETL[RT]. This is SeETL[RT] V 1.6.1.

SeETL[RT] Version 1.6.1 contains the following enhancements:

Memory Mapped IO for Windows 2000+

The parameter `dwh.ctl_dim_table_load_control.load_control_option` can now be set to '4'. This indicates that the dimension table should be loaded into a memory mapped array and that the attribution process should use the memory mapped IO file which will be in a directory pointed to by the SeETL [RT]TEMP environment variable. (The default is "D:\TEMP") This means that many attribution processes using the same large dimension table can all use the same copy of the data in memory thus providing the same high performance of binary search in memory while only requiring one copy of the lookup table for all running attribution processes. This introduces 'unlimited scalability' to SeETL[RT].

The implementation of memory mapped IO for unix systems is planned to follow.

When memory mapped IO for unix is implemented it will be possible for the user to run as many processors as the user would like for the attribution process without paying the overhead of memory. This will make SeETL[RT] truly unlimited in it's scalability capability.

Ability to turn off AutoCommit for ODBC Drivers that support Commit Processing

In previous releases the Data Transfer Utility inherited the AutoCommit properties of the ODBC driver it attached to. However, nowadays nearly all ODBC drivers support the ability to perform Commits without destroying SQL Cursors. For such ODBC drivers it is now possible to turn off the AutoCommit property of the ODBC driver and to specify the commit frequency against the database. The commit frequency is specified through the CommitFrequency parameter and can be set as your DBA recommends.

Testing has shown that on Oracle and SQL Server on Windows 2000 the ability to set the commit frequency to 10,000 has led to doubling the throughput of the Data Transfer Utility when performing large numbers of inserts for fact tables. This increase in performance reduces the number of cases where it is necessary to build Load Image Format Files.

Improved Logic for 'By Name' Data Movement

Because SeETL[RT] is being used on much larger volumes than originally expected the processing for moving fields from a source to a target became an area that consumed a large amount of processing time. This movement of data between sources and targets has been improved. The result was a 25% overall elapsed time improvement when processing large volumes of data.

This change was of particular value in the creation of the Load Interface Files. Load Interface Files are only required when the volumes were so large that the Data Transfer Utility run times would be slower than what is desired. However, the very act of moving the data to the Load Interface File Format itself was also very CPU intensive because this is all the processing the Data Transfer Utility performs when creating a Load Interface File.

The improved logic for 'By Name' data movement improved the processing speed of the Load Interface File processing by a factor of 4. This means it is now possible to produce **REALLY LARGE** load interface files in very fast processing times.

Improved Logic for Extraction of Data

The logic used to extract data from an ODBC source for the programs that read data directly from tables (Data Transfer Utility, Attribution, Dimension Table Processing) has been improved. The previous versions retrieved a row of data using one ODBC call and then retrieved each field from that row one after another using subsequent ODBC calls. This logic has been changed to retrieve the entire row in one ODBC call and not require the retrieval on a column by column basis. This has improved the speed of reading data from an ODBC source between 5-10% depending on the number of columns being retrieved.

The overall performance improvements have resulted in the Data Transfer Utility now running **250%** faster than the 1.5.5. release and the overall performance of SeETL[RT] has been improved by **30%.**

(The attribution process was always written very efficiently because it is the major consumer of CPU in a batch run.)

Ability to Create Load Interface Files During Attribution

In previous releases the attribution programs (AT01/AT02) produced a file which was in the Self Describing File Format. This format of data could then be passed to the aggregation process or directly to the loading process for detailed records. This also allowed for multi-level tables. However, in the cases where multi-level aggregates are not calculated (for example Sybase IWS data model) and the volumes are very large some time can be cut out of the overall processing of records by having the attribution process create load interface files. This means that the output of the attribution process can be loaded directly into the data warehouse where it is known that all records are inserts. This is often the case in very large volume data warehouses like telco CDRs or web logs.

6.2. Non Performance Related Enhancements

As more and more people use SeETL[RT] we are being asked for more and more utilities to make it even more productive and useful. And this is great. The suite of classes makes it very easy to add new utilities. It is really a case of 'the more requests the better'.

DataStage Job Submission Utility

SeETL[RT] is now being used in a number of DataStage sites. It is used mostly as a prototyping tool. However, in a number of sites the Job Scheduling Utility is being used to schedule DataStage jobs because of the superior failure management of the Job Scheduling Utilities. I initially released a simple DataStage job scheduling utility which would just submit a job. However, I have added a new utility to provide the ability to start a DataStage job with parameters. If the user would like to pass parameters to the DataStage job these parameters must be presented via a table/view.

MetaData Checking Utility

SeETL[RT] is 'Typeless' and it does not check data types at run time. This is one of it's great features. If a data type can be converted to the target type by the ODBC driver it will be converted. However, experience in using SeETL[RT] has shown that one of the more difficult 'bugs' to find is where a field of the incorrect data type/contents is sent to another field. To help find these errors the MetaData Checking Utility inspects all source and target columns by name and issues messages where it detects that there might be a problem.

MetaData Printing Utility

One of the real 'missing features' of most databases is that it is not easy to query their catalogs to determine the data types of columns in views. This is quite frustrating. SeETL[RT] makes extensive use of views and often times it would be very handy to just have a printout of the data types of the views. The MetaData Printing Utility prints the metadata for a single table to a single work file in the self describing file format used for other data. It is easy to read and has come in very handy.

MetaData Loading Utility

The MetaData Printing Utility proved such a hit the next question I was asked was whether I could create a utility to load the MetaData for a view into a table, rather than just print it. Of course, this was trivial and so the MetaData Loading Utility was created. This utility allows a user to load the column definitions for a view/table into a working table. The working table defaults to the name **ctl_column_defs.** However, the user can load the MetaData to any target table that meets the format of the table defined in the documentation.

7. APPENDIX 6 – ENHANCEMENTS FOR 2.1

Having introduced massive performance improvements in the 1.6.1. version the point release for 2.1 has focused back on 'functionality' rather than on 'performance'. The major performance enhancement of 2.1 is the implementation of memory mapped IO for unix versions of SeETL[RT].

Unix Implementation of Memory Mapped IO

The introduction of Memory Mapped IO into the unix version of SeETL[RT] is something of a milestone for our product. Simply put, it means that SeETL[RT] can now be used for the most massive clients with the most massive processing volumes occurring today.

This milestone is achieved because the major limitation of any dimensional model ETL process is the size of the in memory lookup tables used to translate real keys into integer keys. If one copy of these tables is required for each 'attribution process' (let's say N copies of the attribution process) running then the volume of memory required is of course N times the volume of memory required to load one copy of the dimension tables into memory.

The ETL vendors have attempted to use memory mapped IO to get around this problem to a greater and lesser level of success depending on which ETL tool you are talking about.

SeETL[RT], by design, implements memory mapped IO in the most effective possible fashion. That being loading the required data into a single set of memory mapped images which are then accessible by all attribution processes and all completely at the users control 100% of the time. No-other ETL product provides such detailed control of the use of memory mapped IO.

The implementation of memory mapped IO also allows the user to put the 'T' portion of 'ETL' onto a machine that does not require a database license. The dimension tables can be loaded across an IP link from the data warehouse itself and the source tables can also be fetched across an IP link. Combined with the fact that SeETL[RT] is available as a source code license it means that the use can place SeETL[RT] attribution processes on a separate physical machine which can be as large as the client chooses without incurring database licenses or extra 'ETL tool' licenses. In effect, the massive processing requirements of attribution processing on very large volumes of transaction can be implemented with very, very low software cost. This makes SeETL[RT] a viable solution even for very large companies who are already using vendor ETL tools because SeETL[RT] can reduce the software costs of performing the ETL processing for the single most expensive portion of Data Warehouse integration processing.

We see this as a very exciting possibility for our clients.

Ability to Set Load Control Option for ALL dimension tables

The Load Dimension Tables into Memory Maps utility has the built in capability to load all tables that are specified as being required to be loaded into memory maps into those memory maps. However, the attribution program CTLAT01 still required that every fact table/dimension table combination be specified in order for the attribution program to understand that the specific fact table required was already loaded into memory. This could be expressed in a single record by changing the processing of CTLAT01 to check for a record representing 'ALL' fact tables. Thus a feature was added to allow the fact table name in the 'dim_table_load_control' table to be '*ALL* to specify that the dimension table was to be loaded into memory for ALL fact tables. As a further result of this enhancement it is possible to specify the load control option to be the same for all dimension tables for a specific fact table saving the extra entries and any possible errors in the dim_table_load_control table.

Ability to Group Dimension Tables Being Loaded into Memory Mapped IO

The table dim_table_load_control now contains a new column called dimension_table_group. The column dimension_table_group allows the user to parallelise the loading of dimension tables into memory mapped IO into specific groups. This allows the user full control over the grouping and loading of dimension tables into memory mapped IO to support the maximum level of efficiency of processing.

Please note, installed clients must alter their ctl_dim_table_load_control table to contain this new column.

Ability to Set a Specific the Memory Mapped IO Directory Using a Parameter

In the initial windows version of the SeETL [RT] the memory mapped IO directory was specified in the environment variable $ SEETLTEMP and if $ SEETLTEMP was not specified the directory used was D:\TEMP.

On the initial solaris implementation of memory mapped IO my solaris advisors recommended that the memory mapped IO files always go into /tmp. However, when we tested the memory mapped IO on AIX my AIX advisors told us that on AIX /tmp does not mean anything to AIX and that the directory used for memory mapped files should be passed to SeETL[RT] as a parameter for AIX implementations.

From this feedback we have created a new parameter called MemoryMappedIODirectory. The MemoryMappedIODirectory parameter can now be used to pass the directory to be used for memory mapped IO files for all operating systems that SeETL[RT] runs on.

Support for MySQL

Over the last few years MySQL has been making inroads into the lower end of the relational database market. It has become almost ubiquitous as the database to support small php data driven web sites on linux and unix. MySQL claim more than 5 million users of the product. In recent times a number of clients have also asked us if we have any plans to support MySQL. Our clients see it as an 'interesting possibility' because of the 'price'. We had always taken the position of 'not supported' with MySQL 4.x because it was missing some fundamental features we required, such as views.

With the release of MySQL 5.0 MySQL AB has included a number of features to support data warehousing. These include enhancements to the optimiser, views, stored procedures and advanced data management. We have decided that MySQL is now mature enough for many of the clients and prospects that we have and we have introduced MySQL support in this release. With SeETL[RT] 2.1. we are committing that MySQL will be supported into the future to the same extent as the other databases currently supported.

Data Correction Utility

One of our largest clients asked us to provide code to correct and reformat a large amount of files (100+) they wanted to place into their data warehouse. We wrote the code and decided it was worth making into a new utility for other clients. Further development on this utility is expected so feel free to send us your requests at support@instantbi.com.

Introduction of Report Services Reports

As we develop a revolutionary new product we have started using SQL Server Report Services. We have decided to release a suite of Administration and Metadata reports in Report Services. The examples of these reports are contained in this appendix.

Introduction of Commit Processing for DM01, DM02, AG01

The previous release of commit processing only included the Data Transfer Utility since it processed by far the largest data volumes. The DM01 and DM02 programs only processed large volumes for the initial load of data for large clients so commit processing was not as important.

However, one of our largest clients intended to perform a significant number of 'Day 1 Startup' tests on large volumes of customers and requested that we added commit processing to DM01/02. So we did. We also decided to add commit processing to AG01 just to make all programs that update any significant volume of data perform commit processing.

Thus, the parameters 'AutoCommit' and 'CommitFrequency' are now valid parameters for:
CTLU001 – Data Transfer Utility
CTLDM01 – Type 1 Dimension Processing
CTLDM02 - Type 2 Dimension Processing
CTLAG01 – Aggregation Processing

Additional Metadata Mismatch Checks

Additional Metadata Mismatch checks were added to the Metadata Mismatch checking utility.

The specific tests and error messages added are:

CTLS0802_MSG	Metadata mismatch detected. The source field is not a DATE field and the target field is a DATE field. This may cause abnormal termination of the ETL processing. The source and target fields are as follows:
CTLS0803_MSG	Metadata mismatch detected. The source field is not a TIMESTAMP field and the target field is a TIMESTAMP field. This may cause abnormal termination of the ETL processing. The source and target fields are as follows:
CTLS0804_MSG	Metadata mismatch detected. The source field is some kind of character or binary field and the target field is some kind of numeric, float or integer field. This may cause abnormal termination of the ETL processing. The source and target fields are as follows:

Introduction of NumberRowsToTransfer Parameter to AT01, AT02, PR01, AS01

The NumberRowsToTransfer parameter previously only applied to the Data Transfer Utility. However, a very large client asked for us to include it on the attribution and profiling programs for a Sybase IWS implementation. Since we were asked for these additions by a large client we decide to add the parameter to all the programs that is makes sense to apply the parameter to.

By being able to limit the number of rows processed using a parameter it is easier to test against very large volumes of data. We used to include a constraint on the view, now we can just include a parameter on the command that invokes the job.

Ability to Restart a Partially Abended Batch

With the unlimited scalability of SeETL[RT] we discovered the 'minor inconvenience' that when we had very large and complex batches and a portion of the batch failed that we had to wait until the remainder of the batch completed before we could restart the batch, or we could manually restart the Process Group that had failed.

We decided this inconvenience should be removed. So in release 2.1 we updated the Scheduler to be able to restart a partially failed batch. That is, if one or more process groups inside a batch fails it is possible to simply 'Restart' the batch and the scheduler will restart just those process groups in the batch that have failed.

It does this by understanding which processes are running and which are abended and which have been successfully completed. There are warning messages generated for the process groups that the scheduler attempts to restart which have continued running.

As a part of this upgrade the Process Group Log and the Process Log no longer record all invocations of each process and the final status of that invocation. The logs now record just the one invocation of any given process or process group and the scheduler now performs updates on these rows to describe the current status of the process group and processes.

Ability to Imbed Commands into the SeETL[DT] Mapping Spreadsheet

As part of our overall goal of moving everything that can be moved into the SeETL[DT] spreadsheet we have decided to provide the ability to move commands to be executed into the SeETL[DT] spreadsheet. The commands are then placed into the database rather than into the operating systems file system. This simply removes the need of the user to place the commands into the file system or into the path of the userid under which the SeETL [RT] runs.

The scheduler then read the commands from the table ctl_process_commands and submits them to the operating system. The parameter character used is '?' to make it different from both unix and windows so there is no confusion as to the fact that the field is a parameter.

We have introduced the ability to have 9 (nine) parameters passed as ?1, ?2, ?3 through to ?9. The same parameter can be placed into a command in many places. It also allows the user to pass some parameters to the command using the SeETL [RT] Scheduler and then pass still more parameters to the final command that might be run using the % or $ parameters of windows and unix systems. Any % or $ characters will be passed directly through to the operating system and not changed in any way by the SeETL [RT] Scheduler.

If the command the user wants to execute is simply a system command then he/she should enter the command as normal into the batch commands portion of the batch schedule in the mapping spreadsheet. Any commands that are not found in the ctl_process_commands are assumed to be in the path and will be submitted as normal.

We thought it would be 'nice to have' this capability.

We will continue to place more and more information into the SeETL[DT] mapping spreadsheet.

Ability to Specify Commands to Execute on Failure of a Command

In previous versions the Scheduler has not been able to issue some sort of 'fail command' in the event that a command fails. The recommendation has been to issue a 'success' command, or a series of 'success commands' during processing so that the support staff of the Data Warehouse can know that processing is occurring as expected.

A number of our clients asked us to include the capability to execute a command with a parameter defined for that specific command in the event of failure of any specific command. So, in this release we have finally included this feature!!!

The ctl_commands table now carries two extra columns:

```
,process_program_name_fail    varchar     (255)       not null    default 'unknown'
,process_program_parms_fail   varchar     (4000)      not null    default 'unknown'
```

These columns are set to the command and the parameter that should be executed in the event of a failure of the command specified in the columns:

```
,process_program_name         varchar     (255)       not null    default 'unknown'
,process_program_parms        varchar     (4000)      not null    default 'unknown'
```

By 'failed' we mean that the command in `process_program_name` returns a non zero return code. We have not implemented conditional processing based on varying return codes and we currently have no plans to do so.

We recommend care be taken when specifying commands to run on the failure of a command. The main reason is that the cause of the failure for the initial command may well stop the 'fail command' from processing correctly as well. We still recommend that some form of 'success' messages are sent to the support staff for the data warehouse so that they know that processing is progressing as expected.

We do not recommend that our clients absolutely rely on the 'fail commands' to detect failures because of the possibility that the reason for the failure may also stop the successful processing of the fail command.

Note1: The command specified in the `process_program_name_fail` field will not be looked up in the ctl_process_commands table. Commands specified in the field `process_program_name_fail` are expected to be visible in the path of the userid that is running SeETL[RT].

Please note, installed clients must alter their ctl_commands table to contain these two new columns.

Start and Stop Timestamps added to Audit Table

Up until now there has not been that much discussion/concern over the number of rows processed per second using the SeETL [RT]. It has always been 'fast enough' for clients in the 'under 500,000 customers' and under 1M transactions per day range. This included all of our early clients as we were only marketing the SeETL [RT] to smaller companies.

However, with SeETL [RT] 2.1 and the introduction of memory mapped IO on unix platforms the question of 'how many rows per second per processor' has been asked by a number of large prospective clients wanting to know the higher end scalability of the SeETL [RT].

Indeed, we have been testing with 10M rows in a customer table and many dimension tables in excess of 4M rows.

It has always been possible to calculate the rows per second from the information in the message table (start and stop timestamps) and the information in the audit table (number of rows processed).

However, some clients have asked if we could put both the start and stop times onto the audit record. Of course, this is simple and now the audit table has changed to include started_tstamp and ended_tstamp.

Please note, installed clients must alter their ctl_audit table to contain these two new columns.

7.1. Introduction of 'Push Button ETL Generation' SeETL DesignTime

We have saved the 'best until last'. If you were reading this 'upgrade list' you would be justified in thinking 'what have you guys been doing for the last 5 months?'

The answer is

Revolutionising ETL development beyond all recognition!!!

With SeETL^{RT} 2.1 we are releasing SeETL^{DT} V1.0.

Please allow us to explain.

Every data warehouse project has at least one very time consuming, unavoidable, complex, and critical piece of work. This is the piece of work that documents the mapping of source fields to target fields in the data warehouse and the transformation that must take place during the processing of the data from the source system to the target system.

Today, there are still no good tools to support this process. The humble spreadsheet is still the most used tool in this specific area. Indeed our consultants, and the consultants of our colleagues around the world use spreadsheets on a daily basis to document the source to target mapping for their BI projects.

After the source to target mappings are documented in the 'mapping spreadsheet' each and every data warehouse project must convert that documentation into some form of 'executable code' in some kind of suite of tools such that it can actually be run.

This very action of converting the documentation to 'code':

1. Breaks the relationship between the documentation and the code requiring dual maintenance.
2. Means that forever after the 'code' and the documentation will never be kept in sync because programmers just do not update the documentation from which they write code.

Since we cannot live without the mapping spreadsheet the question that has been on our collective mind for years and years is:

Is it possible to generate all the ETL from the mapping spreadsheet?

Now, with SeETL^{RT} as the ETL tool the answer to this question has become 'Yes'.

The software component of SeETL^{DT} is a VB.NET application that reads the SeETL^{DT} Spreadsheet and generates 95%+ of all ETL code required to build a data warehouse. ***95%+!!!*** And we are working on the other 5%!!

From our research and testing we know that it is possible to place ***ALL ETL*** code requirements into a single spreadsheet and to generate ***ALL ETL CODE***. As we said, we have been busy revolutionizing ETL development beyond all recognition!!!!

SeETL^{DT} is a flexible and extensible utility. It has been written to be able to perform the following functions:

1. Read and worksheet in an Excel workbook.
2. Extract the data cells from the workbook and place them into an internal structure for later use.
3. Use the data in the internal structures to perform processing according to the processing instructions provided by the user in the processing worksheet.

With this flexibility developed into SeETL^{DT} from the very beginning the following is clear.

1. The SeETL^{DT} Spreadsheet can capture and store in it's internal structures any data that can be expressed as a cell in a worksheet.
2. The SeETL^{DT} Software is only limited in what it can 'generate' from such data by the language it is written in, that being VB.NET.

These two statements mean that anything one of our clients would like to add to SeETL^{DT} to be generated as part of their data warehouse project can be generated if:

1. What is to be generated can be expressed as a cell in a spreadsheet.
2. The code to be generated can be programmatically expressed in VB.NET as a function of the input data and knowledge built into the VB.NET code.

We believe that this is pretty much everything. As we have reviewed our development plans for this utility we have not found any source code in any of our recent data warehouse projects where we could not store data in a spreadsheet and generate the code we need.

Thus far the only piece of code that we question whether it is worth putting into the mapping spreadsheet are function definitions and stored procedures. Everything else can go into the mapping spreadsheet.

SeETL^{DT} provides the ability, for the first time, in the history of Business Intelligence, to store all the metadata required to build the data warehouse in one place and to generate all code for all components from that one place.

A revolution indeed!!!

8. APPENDIX 7 – ENHANCEMENTS FOR 3.0 BETA

With the rollout of SeETL 2.1 and the realisation that the software was fast becoming an end to end ETL tool we decided to make a number of key changes to SeETL 3.0. The main change of which was the total rebranding of the product and the renaming of all the objects in the software. A major change indeed.

Instant Data Warehouse has gone and SeETL has arrived. This is to reflect the more mature and broad nature of the product. It is no longer just a product that will load data warehouses, though it does that very well. It is now a generalised tool for Extract, Transforming and Loading data from many sources into ODBC compliant databases.

Renaming of All Objects

With SeETL 3.0 all programs, all files, all parameter, all tables have all been renamed to the SeETL naming standards. This will require a small migration effort from existing clients. To assist in this migration effort all IDW 2.1 code will run on top of SeETL 3.0. It is expected that SeETL 3.1 will 'make the break' from IDW 2.1.

SeETL 3.1 is currently scheduled to be available around mid-year 2007.

Vastly Increased Functionality of SeETL DesignTime

SeETL DesignTime is constantly having new worksheets put into the workbook to define some useful data for ETL. As a result, numerous new worksheets are added and numerous new reports are being written to document those worksheets and publish them to the web.

Ability to use Sequence Numbers on Fact Tables

I have long wanted to add this feature and now it is finally 'in' SeETL. Many data warehouses we have implemented use sequence numbers on fact tables. And many clients want them managed outside the database. We have, to date, not done this as we need to be able to manage the communication between many instances of the attribution process to correctly allocate the keys such that there is no re-use of a sequence number. This is not a trivial addition to SeETL and because sequence numbers are readily available in most databases the feature just didn't make I to the top of the list.

As in many cases, we had a client that really needed sequence numbers and really did not want to put them into the database, and so we wrote in sequence number support for fact tables.

The sequence number support is smart enough to lock the sequence number table while it is getting the next batch of numbers so that sharing is fully supported. It also allows the Architect to define how many numbers will be allocated in each batch of numbers so that he/she can control how often the attribution process must stop and fetch new sequence numbers. The recommended number is 10,000 integers at a time, however, on extremely busy systems with extremely large volumes the number the Architect decides to use may be significantly higher.

Please note, it is recommended that the database not be set to repeatable read for the allocation of sequence numbers to work correctly.

9. APPENDIX 8 – ENHANCEMENTS FOR 3.0.00 RELEASE VERSION

We have been busy on a number of fronts and we have let SeETL 3.0.00 settle through an extended beta process. We are finally ready to release SeETL 3.0.00 as a release version and this appendix documents the extra features found in this release over and above the beta documentation in Appendix 7.

Ability to Send Error Messages and Audit Messages to a Single DSN connection

A number of clients using SQL Server have decided they would like the staging area and the data warehouse to be in different databases. This has meant that error messages and audit messages were recorded in each of the databases. This was seen as a 'nuisance' and it was requested that the ability to send error messages and audit messages to the one database connection be included.

The new parameters to do this are:

- DBConnectionMsgParameter
- MsgCatalogName
- MsgSchemaName

If these parameters are not present the normal behaviour of connecting to the 'Out' parameter equivalents will occur.

Ability to turn off Table Validation

We had originally included a parameter to turn off table validation. We never did implement it in the code because the cost of performing the table validation was so small. With the advent of BI4ALL and how the SQL Server Optimiser resolves the plans we decided to actually implement this feature. Therefore able validation can now be turned off at a program level.

For table validation we also introduced the where clause 'where 1=0' on all table validations. This allows the optimiser to understand that no rows will be returned and to just prepare the statement to make sure that the table itself is valid. This eliminates even the small amount of time spend materialising the view. Very much a small 'nice to have'.

Corrected an Error in CTLDM01/02 Committing

The programs CTLDM01/02 both had an error in committing transactions which caused the loss of the 'last key used' update at the end of the programs. This was corrected so that the update to the ctl_last_key_used table is also committed.

Initial Load Programs for Large Dimension Tables

SeETL is designed to be multi-level from the ground up. One of the design decisions in doing this was that dimension tables would need to be updated during processing because the higher level keys must be inserted onto the detail record when the detail record is inserted. This means that when performing 'day 1 start up processing' the creation of the customer and account dimension tables can take a long time.

However, many clients are not designing multi-level databases. Therefore this feature is not really needed in these cases when performing 'day 1 start up processing'. To speed up 'day 1 start up processing' for clients who do not design multi-level databases we have delivered 'cut down' versions of CTLDM01/02 that send their output to either a self describing delimited file or to a Load Interface File. By providing these cut down versions of CTLDM01/02 clients who have large dimension tables to load who are not designing multi-level databases can take advantage of the performance improvement for 'day 1 start up processing'.

These programs are called CTLDM11 and CTLDM12.

These programs cannot be used for subsequent incremental processing.

Allowing Multiple Blanks in Command Processing

Yes. Someone found that we had a bug in the command processing such that when multiple blanks were put into the command parameters the scheduler determined that the parameter was a zero length character string. So we now remove multiple blanks in parameters before they are passed to the command processor.

Audit Number of Records Discarded in Association and Dimension Table Processing

In the Association and Dimension Table processing SeETL determines if a row has been sent to the Data Warehouse which is a duplicate of a row already in the Data Warehouse. In most processing SeETL attempts to determine if the user has asked for redundant processing and it then avoids that redundant processing.

However, when auditing the results of a run there is no record of the number of rows that have been discarded. So we have introduced audit messages to log the number of rows that have been discarded because they would have been redundantly processed.

Introduce the Order By Clause for CTLU001

To our great surprise MSFT and SQL Server do not guarantee that an order by on a view will be respected in the extraction of the data from the view if the view is merely opened with select * from <viewname>. It happens to work all the time but we found a tech note from MSFT to say there was no guarantee that it would work.

In order to make sure that the DTU is able to return rows in the order expected we introduced two new parameters for DTU. They are:

- UseOrderByClause
- OrderByClause

When an Order By Clause is desired the user can set UseOrderByClause to Yes and then specify the order by clause in the OrderByClause parameter.

This feature is required when performing delta detection. In the delta detection processing the data must be sent to the delta detection utility in primary key sorted sequence where the keys are considered characters.

Introduce the NumberRowsToTransfer to CTLDM01/02

Customers with large dimension tables had placed top nnnn rows or some constraint of the selection of the dimension table rows when performing testing. One of our large clients asked us to add the NumberRowsToTransfer parameter onto our dimension processing so that these views would not need to be changed when wanting to perform day 1 start up processing.

So now you can specify the NumberRowsToTransfer on both CTLDM01/02/11/12.

Introduce the Ability to Validate and Default Invalid Dates

Dates, dates, dates…..they are the bane of the Data Warehouse Architects life. So many people think it is a good idea to invent a different format for a date other than the ISO format. We support another 10 or so date formats now. We also introduced the ability to validate a date and if it is found to be invalid to default it to some parameter passed to the program. This has been done in our Data Correction Utility CTLU016.

Introduction of TurboRead Parameter

SeETL reads fields from the returned result set from ODBC one field at a time. It is possible for ODBC to return the entire row in one call, or even an array of rows in one call. We have coded and tested the routines to return a full row in one call. However, we have found that not all ODBC drivers we have tested will support this calling mechanism so for some years we have not included it. The other reason for not including this feature was that other performance improvements across SeETL made the implementation of this feature very marginal.

Now we are well into the maturing stage of SeETL we are going back and putting in the 'marginal' improvements where we think it is worth the effort and this is one of them.

It is now possible to set 'TurboRead=Yes' on most programs that may process large amounts of data coming in from a table and this will provide a 10% or so decrease to the overall program processing time.

Performance Improvements for Dimension Processing

Customer needs are the driver for the implementation of change to SeETL and this is a great example of this. We had a client who has a business process of re-scoring their 14 million addresses each quarter and they wanted to retain the new set of scores each quarter. This required an archive table (implemented as a type 2 dimension) that would require a process that would place a complete new image of the table into the archive each quarter. To make matters worse there are nearly 200 fields on the archive.

Dimension table processing was built to be incremental and it even checks all fields for change to see if the record has actually changed. Thus, the dimension table processing was not suitable for the solution the client wanted to implement.

In response we developed the 'initial load' dimension table processing and we also introduced a suite of improvements inside the dimension table processing such that for very large numbers of dimension table processing is sped up by more than 95%.

This improvement is only of concern to those customers that have very large numbers of fields on rows or very large numbers of dimension table rows that are changing on some regular basis. However, for these customers, it will be very useful.

Introduce Order By Clause for CTLAT01/02

When using clustered tables in databases like SQL Server it was found to be very beneficial if the data that was presented to the loader was presented in clustering sequence. In other databases the order by clause could be placed onto the view that CTLAT01/02 read to produce the data to be attributed. Unfortunately, SQL Sever, though it does present the data in the order requested, does not guarantee that it will do so in all cases. To guarantee the order in which the data to be attributed will be presented to CTLAT01/02 we have introduced the parameters UseOrderByClause and OrderByClause to CTLAT01/02.

Introduce Batch Processing Sleep Utility

On windows there is no standard utility that allows you to sleep a batch for a specific period of time. There are utilities that can be downloaded and integrated into the schedule. A client asked us for a program that would allow them to put a sleep into the batch using standard utilities and so we developed this utility.

Introduce XML File Reformat Utility

The amount of XML based data is increasing. And companies are starting to need to load XML data into their data warehouses. There are a lot of XML covenrters out there and most of our clients have been satisfied with using an XML converter and then reformatting the data to be loaded. This has required the use of a third party XML converter. We were asked by a client if we could write a relatively simple XML converter to convert XML data to the SeETL internal file format.

Introduce Target Rows Delete Utility

The Data Transfer Utility has an option to be able to delete rows that already exist prior to insertion. We also introduced the ability to delete rows while building a Load Interface File. We then introduced the ability to respect sequence numbers for rows for fact tables.

With the introduction of the ability to respect sequence numbers we introduced the issue that if the Data Transfer Utility was run and was selecting sequence numbers of rows and then those rows were being deleted and the program failed there would be no way to get those sequence numbers back.

In short, the Data Transfer Utility was not re-runnable from the beginning when creating a Load Interface File and deleting the rows that previously existed. We introduced the possibility that users could lose data if they exercised this set of options.

Hence the Delete Target Rows Utility was needed to make the entire process re-runnable. The Data Transfer Utility can write a file for rows to be deleted. This utility can then be run to delete the rows. Therefore each step is restartable from the beginning.

By doing this, we now enable the user to create a Load Interface File, sorted in clustered sequence, that respects the previously used sequence numbers with no loss of data with all steps being re-runnable from the beginning of the step. And all simple and easy to do, unlike trying to do the same in any other tool.

Introduce Bulk Load Rows into SQL Server Utility

The suite of SeETL utilities has been designed to be database independent. The specific databases implement things like the loaders slightly differently and so we did not write to the bulk load interface for the various databases. This works fine for all databases except SQL Server, and only when using partitioned clustered tables.

When using SQL Server with partitioned clustered tables the performance of the deleting and loading is relatively poor compared to other databases, as well as compared to SQL Server when the tables are not clustered.

So, as we were working on larger SQL Server databases we decided to write a specific utility to take advantage of the bulk loader for SQL Server as well as taking advantage of performing bulk operations on the clustered tables so that the work could be pushed into the database engine rather than being executed on row at a time as per normal.

This utility was specifically written to improve the load times for SQL Server for partitioned clustered tables.

It has the added benefit that it reduces the number of steps in the schedule.

Introduce Execute ProcessName from Scheduler Utility

When testing SeETL jobs we have always created command files and executed those command files. Once the processing was tested we then embedded the commands into the scheduler and ran the commands from inside the scheduler from the ctl_process_commands table.

This all works fine with the exception that when one makes changes to the commands inside the SeETL workbook one has to also remember to make the same changes to the test commands. If one forgets then occasionally the testing results do not property match the results produced by running the scheduler.

One of our clients asked us if we could develop a command where they could issue a request to run a command from the ctl_process_commands table in the target database from the command line.

The way this was developed was to enable the passing of parameters into the command as well as the ability to over-ride the parameters that are not adjustable inside the command by merely appending them to the command that is entered at the command line interface.

Make The Scheduler Robust Against Network Failures

SeETL is actually intended to be running on the machine that the DW database is on. Because most clients buy the ‘source code’ version and not the one machine runtime license there is no reason to try and put SeETL on one machine and the databases on other machines.

However, in recent times we have had some clients that want SeETL installed and supported on one single ‘ETL machine’ and they want their staging and target databases installed on two other machines making three machines in the setup with 1GB Ethernet between them. Ok..the client is always right!

In some of these cases the network suffers transitory failures. Very brief outages that no-one has really noticed before. Until now. The scheduler is sensitive to such failures since it checks the schedule every BatchSleepSeconds and if there is a transitory network outage it considers this an error. Not only that, the error is usually such that it cannot write to the error log either!! So the symptoms seen initially just looked like ‘the scheduler went away for some unknown reason’.

We have updated the Scheduler to detect network outages in SQL Server. (When we need to we will also test on other databases.) When detected the Scheduler will consider it a ‘soft error’, sleep for BatchSleepSeconds and then try again.

Updating Dimension Table Last Key Used Inside Commit Processing

The client that has the transitory network outage also discovered one of very few bugs. The dimension table processing once suffered the same fate of a transitory network failure. And we discovered that the update to the Last Key Used in dimension table processing was not performed at the same time as the commit for the rows written to the database, it was only written at the end. This caused a problem with keys. Se we have updated DM01/02 to update Last Key Used inside the commit processing for each batch of records.

SeETLRTE00011 - Include the ability to unload fields containing only blanks

Yes. We found a client that has fields that contain all blanks and the all blanks actually is meant to mean something. Really, we are not kidding! So they wanted the ability to unload and load data where, if the field was entirely blank, then the blanks would be unloaded. In the spirit of 'the customer is always right, even when the request is clearly just a little crazy, and the customer even agrees" we added a TruncateTrailingBlanks parameter and defaulted it to Yes to maintain existing behavior.

SeETLRTE00012 - Delimited File Reformatting – Support Delimiters Inside Quoted Strings

Yes. We found a client that has delimited data coming into the data warehouse where quoted strings contain the delimiter character inside the quoted strings. Needless to say our utilities for reformatting delimited files did not handle this situation. So we have added the ability to handle this. We wonder if we will ever see it again!

SeETLRTE00013 - Delimited File Reformatting – Support Quotes Inside Quoted Strings

Yes. We found a client that has delimited data coming into the data warehouse where quoted strings contain quote characters inside the quoted strings. Needless to say our utilities for reformatting delimited files did not handle this situation. So we have added the ability to handle this. We wonder if we will ever see it again!

SeETLDT00001 - Allow Current Flag = 1 for type 1 Dimensions

Type 2 dimension tables always have type 2 processing. Right? Wrong. We have found some interesting uses for allowing a table to kind of 'morph' between being processed as a type 1 and as a type 2 table. Meaning that there are some fields that we only want to replace when they change while not having to test all the other fields for change. It was kind of a 'neat trick' which we have only just seen for the first time.

To enable this 'neat trick' to happen it was necessary to be able to generate views over the type 2 dimension tables to set 'current flag = 1' to 'morph the dimension table from type 2 to type 1 for the specific piece of processing.

SeETLDT00002 - Add multi-level support for dimensions

Those who know Mr. Nolan know that he has been talking about multi-level data since he first learned about it in 1994. Alas, the 'perceived need' for multilevel data has gone away and few people know why it was a good idea any more. In this release we have added in the ability to define processing in the SeETL workbook for multi-level data.

Introduce Allow Data Conversion Errors in Loading

"Invalid character for cast specification". This innocuous message is the most frustrating one we have. It means that the ODBC driver cannot put the data into the field because the data type does not allow the conversion. However, ODBC does not tell us **WHICH** field or **WHAT** data. To get around this error in testing we introduced the parameter AllowDataConversionErrors=Yes which will continue processing in the face of rows with data conversation errors.

An additional piece of functionality was then also added to allow the actual rows in error to be written to a file. The parameter for the file is called ErrorFileName. So now, when a data conversation error occurs it is possible to get the record that caused the error to be written to a file. This allows easier debugging of data causing errors.

Allow Alternate View For Sequence Number Lookup

With the ability to respect sequence numbers on fact records there is an issue. How to look up the previous fact record to retrieve the sequence number to be respected? The initial implementation used the set of integer keys at the front of the fact table as the lookup fields to retrieve the sequence number. However, in the case where any one of these keys change the lookup will fail and the row will not be retrieved correctly. Although it is uncommon for the integer keys to change from one version of a record to another it is far from unheard of.

To get around this problem we have implemented to ability to provide a view name as an alternate view that can be supplied to perform the lookup to retrieve the sequence number to be respected.

Further, to get around the problem of 'how to name the fields' we have also introduced the capability to support 'FuzzyFieldMoves'. The idea of ; FuzzyFieldMoves' is that two field names are considered 'equal' if they are identical except for 'pk_' being a prefix of one or the other field.

By allowing 'FuzzyFieldMoves' and using the alternate view it is possible to define an alternate key using the real key for the record to be able to retrieve the correct sequence number to place on the current record in all situations.

Allow Fuzzy Field Moves

As per the entry for above UseAlternateViewForSequenceNumberLookup the AllowFuzzyFieldMoves also has broader implications when used for the Data Transfer Utility. When moving data from a source to a target inside the Data Transfer Utility the AllowFuzzyFieldMoves parameter allows the DTU to consider two field names equal if the only difference between them is that one of them starts with "pk_". This allows such things as being able to move data from a file/table that does not have the keys defined on it to a target table that does.

The introduction of AllowFuzzyFieldMoves allows for less coding of views as delimited files or source tables bring data into the SeETL environment.

10. APPENDIX 9 – ENHANCEMENTS FOR 3.1.00 RELEASE VERSION

The vast majority of the enhancements for 3.1.00 have been to the design time side of the product and not to the Run Time side of the product. In SeETL[RT] the major new enhancements have been to provide a more robust version of the scheduler and to allow the execution and logging of SQL statements that are stored inside the database and not in files.

These changes are as follows.

Provide Network Failure Resilient Scheduler

Some clients placed the scheduler and the database that the scheduler reads its schedule from on different machines. When the LAN between these two machines goes down or has an intermittent problem the scheduler loses its connection to the database and the scheduler stops. These clients generally have a single scheduling machine and distributed staging and target databases.

Therefore a second scheduler was provided. This scheduler has had a MANY internal changes such that it connects to the database and disconnects from the database each BatchSleepSeconds when it wakes up to check that the schedule is progressing ok. The prior scheduler remains available and supported. The prior scheduler makes the implicit assumption in all areas of the program that the program connects to the database and remains connected to the database for the life of the program.

Therefore the new scheduler had to have many changes to it to remove this assumption.

This is the reason that there are two schedulers now.

Execute SQL Loaded Into DW Database

The BIG change in SeETL[DT] in 3.1.00 is that the tool is now able to generate SQL code as the ETL subsystem. This means that large chunks of SQL can be executed by the database. The previous utility for executing SQL was only intended to execute small fragments of SQL and then only from files. The auditing was no more than placing the small fragment of SQL into the message log with the timestamps and the number of rows processed.

This logging is not sufficient if the SQL being executed is actually the ENTIRE ETL SUBSYSTEM. Therefore a much improved new utility was required.

The new utility accepts a parameter that is the name of the file that the SQL was loaded from. It searches for the current versions of that SQL Statement loaded into the database, executes it, and logs the start and stop timestamps, the sql code result, the number of rows processed and the first 4,000 characters of the statement that was actually executed.

This provides a FAR more robust, secure and auditable way to run SQL statements. This is because the statement is loaded into the database and all the security and audit features of the database are available which are far superior to the same features of a file system.

11. APPENDIX 10 – ENHANCEMENTS FOR 3.1.01 RELEASE VERSION

The vast majority of the enhancements for 3.1.01 have been to the design time side of the product and not to the Run Time side of the product. In SeETL[RT] the major new enhancement has merely been to migrate to the 64 bit version of Visual Studio and testing to make sure the 64 bit version works properly.

As of 3.1.01 we will no longer support 32 bit versions of SeETL[RT] as well as the 64 bit versions. Windows Server 2008 has been out for long enough now that there is no need to support Windows Server 2003. Further, 64 bit hardware is so prevalent that anyone still running 32 bit databases and 32 bit hardware for data warehouses are very well advised to migrate up to the 64 bit versions.

We have seen great performance improvements for SeETL[RT] on the 64 bit versions.

Logging Process Group Start Events

There has been a new control table created to log the start events of process groups.

```
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'
    )
```

Logging Processids from the Scheduler

There have been new fields added to the process group run log and the process run log.

```
alter table dbo.ctl_proc_grp_run_log add
       parent_process_id      integer           not null    default 0
;

alter table dbo.ctl_proc_grp_run_log add
       child_process_id       integer           not null    default 0
;



alter table dbo.ctl_proc_run_log add
       parent_process_id      integer           not null    default 0
;

alter table dbo.ctl_proc_run_log add
       child_process_id       integer           not null    default 0
;
```

12. APPENDIX 11 – ENHANCEMENTS FOR 3.1.02 RELEASE VERSION

There have been no updates to SeETL Run Time in the 3.1.02 release. We are keeping the release numbers the same to keep them in sync and nothing more.

Once we settle the SeETL Design Time Beta releases then we may add some small changes to SeETL Run Time.

The main difference now with SeETL Run Time is that we no longer need to protect the source code as rigorously as we used to do. Now we have put license keys into SeETL Design Time the mechanism by which we will protect our source code is by the licenses in SeETL Design Time. Because of this we have published the source code of SeETL Run Time in the GA Package so that people can review it in more detail.

We have not released the classes etc. But from the source code the reader will be able to understand how we were able to do what we have claimed to do all these years, a claim that many people denied could possibly exist.

Those people who are in to writing C++ will be interested, from a personal learning point of view, to read the code that we have been using and selling all these years.

We believe that not many people are going to go with the C++ RunTime version in the future. We believe that most people are going to go with the SQL ETL Generation process that is now available in the SeETL Design Time version.

SeETL Design Time passed 70,000 lines of vb.net code in this release. If there is great demand for it then we might convert it to C++ which is the language that it really should be written in for performance and portability. However, as so often happens in IT, SeETL DT started out as an idea that was not meant to be extended very far, and was written in a “convenient” language, vb.net.

Now it has far outgrown vb.net and it would be a good idea to migrate it to C++ at some point in the future. We are far more proficient in C++ than we are in vb.net as you can see from the code that has now been released in the GA Package.

13. APPENDIX 98 – EXAMPLE SEETL^{DT} BATCH SCHEDULE

This section provides an example batch schedule. It is an example batch schedule that was developed for a recent client. The names of the tables have been changed to non-meaningful names. The sample batch schedule spreadsheet is available from the downloads page.

Row Type	Batch Name	Pre Req Number	Pre Req Type	File Name	Day of Week	Day of Month	Start Hour	Start Minute	Start Timestamp
bpr	testbatch001	1	FileExists	/seetl/seetldev/data/testbatch001.run.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch002	1	FileExists	/seetl/seetldev/data/testbatch002.run.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch003	1	FileExists	/seetl/seetldev/data/memorymappedio1.wait.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch003	2	FileExists	/seetl/seetldev/data/memorymappedio2.wait.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch003	3	FileExists	/seetl/seetldev/data/memorymappedio3.wait.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch003	4	FileExists	/seetl/seetldev/data/memorymappedio4.wait.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch003	5	FileExists	/seetl/seetldev/data/memorymappedio5.wait.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch003	6	FileExists	/seetl/seetldev/data/memorymappedio6.wait.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch003	7	FileExists	/seetl/seetldev/data/memorymappedio7.wait.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch003	8	FileExists	/seetl/seetldev/data/memorymappedio8.wait.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch004	1	FileExists	/seetl/seetldev/data/testbatch004.run.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch005	1	FileExists	/seetl/seetldev/data/testbatch005.run.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch006	1	FileExists	/seetl/seetldev/data/memorymappedio9.wait.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch006	2	FileExists	/seetl/seetldev/data/memorymappedio10.wait.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch007	1	FileExists	/seetl/seetldev/data/testbatch007.run.yes	0	0	0	0	1900-01-01 01:01.11
bpr	testbatch008	1	FileExists	/seetl/seetldev/data/testbatch008.run.yes	0	0	0	0	1900-01-01 01:01.11

The batch schedule indicated that testbatch001 will not start until the file /seetl/seetldev/data/testbatch001.run.yes exists. And similarly with testbatch002. However, for testbatch003, the batch testbatch003 will not run until all the files memorymappedio1-8 exist. That is testbatch003 will wait until all the dimension tables into memory mapped files. Then testbatch003 will run and copy the signal file such that testbatch004 will run.

Testbatch005 will then run to load more files into memory mapped io. And then lastly testbatch007 will run and then testbatch008 will run. As a batch ends it copies the signal file is copied to signal the next portion of the batch to begin. Batches and process groups can both be used to create similar types of batches. However, process groups require more connections to the database than creating batches. So we recommend using smaller batches and putting process groups into batches to reduce the number of connections maintained to the database during processing.

Process Groups

Process Groups provide the ability to create parallelism inside batches to make maximum use of the available processors while also avoiding thrashing the machine by simply submitting all processes at once. Process Groups provide the ETL Architect with complete control over the concurrency of jobs. Further, process groups provide the ability to process jobs with dependencies in sequence.

The table below can be read as follows:

In testbatch001 process groups 2,3,4,5,6,7 and 8 are all dependent on process group 1. This means that process group group001 will execute and complete before process groups 2, 3, 4, 5, 6, 7, 8 will start. Then process group group999 is dependent on groups 1, 2, 3, 4, 5, 6, 7, 8. So group999 will not run until all the process groups in the batch have completed successfully. Obviously the command in group999 is usually the command to copy the signal file to the file that the subsequent batch is dependent on. So when a batch ends, a signal file is copied to tell the scheduler that the next batch can start.

In the Process Groups below you can see how the dependencies are created.

Row Type	Batch Name	Group Name	Pre Req Number	Pre Req Type	File Name	Pre-req Process Group
pgpr	Testbatch001	group002	1	ProcessGroup	na	group001
pgpr	Testbatch001	group003	1	ProcessGroup	na	group001
pgpr	Testbatch001	group004	1	ProcessGroup	na	group001
pgpr	testbatch001	group005	1	ProcessGroup	na	group001
pgpr	testbatch001	group006	1	ProcessGroup	na	group001
pgpr	testbatch001	group007	1	ProcessGroup	na	group001
pgpr	testbatch001	group008	1	ProcessGroup	na	group001
pgpr	testbatch001	group999	1	ProcessGroup	na	group001
pgpr	testbatch001	group999	2	ProcessGroup	na	group002
pgpr	testbatch001	group999	3	ProcessGroup	na	group003
pgpr	testbatch001	group999	4	ProcessGroup	na	group004
pgpr	testbatch001	group999	5	ProcessGroup	na	group005
pgpr	testbatch001	group999	6	ProcessGroup	na	group006
pgpr	testbatch001	group999	7	ProcessGroup	na	group007
pgpr	testbatch001	group999	8	ProcessGroup	na	group008
pgpr	testbatch002	group999	1	ProcessGroup	na	group001
pgpr	testbatch002	group999	2	ProcessGroup	na	group002
pgpr	testbatch002	group999	3	ProcessGroup	na	group003

pgpr	testbatch002	group999	4	ProcessGroup	na	group004
pgpr	testbatch002	group999	5	ProcessGroup	na	group005
pgpr	testbatch002	group999	6	ProcessGroup	na	group006
pgpr	testbatch002	group999	7	ProcessGroup	na	group007
pgpr	testbatch002	group999	8	ProcessGroup	na	group008
pgpr	testbatch004	group999	1	ProcessGroup	na	group001
pgpr	testbatch004	group999	2	ProcessGroup	na	group002
pgpr	testbatch004	group999	3	ProcessGroup	na	group003
pgpr	testbatch004	group999	4	ProcessGroup	na	group004
pgpr	testbatch004	group999	5	ProcessGroup	na	group005
pgpr	testbatch004	group999	6	ProcessGroup	na	group006
pgpr	testbatch005	group999	1	ProcessGroup	na	group001
pgpr	testbatch005	group999	2	ProcessGroup	na	group002
pgpr	testbatch007	group999	1	ProcessGroup	na	group001
pgpr	testbatch007	group999	2	ProcessGroup	na	group002
pgpr	testbatch007	group999	3	ProcessGroup	na	group003
pgpr	testbatch007	group999	4	ProcessGroup	na	group004
pgpr	testbatch007	group999	5	ProcessGroup	na	group005
pgpr	testbatch007	group999	6	ProcessGroup	na	group006
pgpr	testbatch007	group999	7	ProcessGroup	na	group007
pgpr	testbatch007	group999	8	ProcessGroup	na	group008
pgpr	testbatch008	group999	1	ProcessGroup	na	group001
pgpr	testbatch008	group999	2	ProcessGroup	na	group002
pgpr	testbatch008	group999	3	ProcessGroup	na	group003
pgpr	testbatch008	group999	4	ProcessGroup	na	group004
pgpr	testbatch008	group999	5	ProcessGroup	na	group005
pgpr	testbatch008	group999	6	ProcessGroup	na	group006
pgpr	testbatch008	group999	7	ProcessGroup	na	group007
pgpr	testbatch008	group999	8	ProcessGroup	na	group008

Commands

The batch schedule and the process groups are used to define the dependencies and order of processes that will run in the batch. That is, these two things control the processing of the commands. The commands themselves are contained in the commands table. A sample is provided below.

As you can see, every command must belong to a batch and a process group. It is also allocated a process step number to force the ordering of commands within process groups. The Process Program Name is the name of the command to be executed. The command being executed can be:

- An operating system command.
- A SeETL [RT] command stored in a file visible in the path.
- A command stored in the process commands table.

The commands below are easy to understand. The batch you might find a little unusual is testbatch007, group999. In this group the signal file is copied to a series of kill files. These kill files are the kill files for the memory mapped io processing. The file names are passed to the memory mapped io commands in the process command table. At the end of attribution processing these programs must be told that they can release the resources they hold. This is done by copying the signal file to the kill files. This is also an example where the memory mapped io files are created in a batch separate to the fact table processing itself. Therefore, it there was any failure in fact table processing there would be no need to reload the memory mapped io files as there would be with some other products. The memory mapped files stay resident in memory until such time as the kill files are created and the memory mapped io processing programs detect the kill files and release their resource.

We have provided a real set of commands actually used at a client so that you can see how a complex batch can be created and many process groups can be made to run in parallel in the batches.

Row Type	Process Batch Name	Group Name	Process Step Number	Process Command Name	Process Program Name	Process Program Parms	Process Program Name Fail	Process Program Parms Fail
bcom	testbatch001	group001	1	DM01	runtype1dimension	dimension_table01	NA	NA
bcom	testbatch001	group001	2	DM01	runtype1dimension	dimension_table02	NA	NA
bcom	testbatch001	group001	3	DM01	runtype1dimension	dimension_table03	NA	NA
bcom	testbatch001	group002	4	DM01	runtype1dimension	dimension_table04	NA	NA
bcom	testbatch001	group002	5	DM01	runtype1dimension	dimension_table05	NA	NA
bcom	testbatch001	group002	6	DM01	runtype1dimension	dimension_table06	NA	NA
bcom	testbatch001	group002	7	DM01	runtype1dimension	dimension_table07	NA	NA
bcom	testbatch001	group002	8	DM01	runtype1dimension	dimension_table08	NA	NA
bcom	testbatch001	group002	9	DM01	runtype1dimension	dimension_table09	NA	NA
bcom	testbatch001	group002	10	DM01	runtype1dimension	dimension_table10	NA	NA
bcom	testbatch001	group002	11	DM01	runtype1dimension	dimension_table11	NA	NA
bcom	testbatch001	group002	12	DM01	runtype1dimension	dimension_table12	NA	NA

bcom	testbatch001	group002	13	DM01	runtype1dimension	dimension_table13	NA	NA
bcom	testbatch001	group002	14	DM01	runtype1dimension	dimension_table14	NA	NA
bcom	testbatch001	group002	15	DM01	runtype1dimension	dimension_table15	NA	NA
bcom	testbatch001	group002	16	DM01	runtype1dimension	dimension_table16	NA	NA
bcom	testbatch001	group002	17	DM01	runtype1dimension	dimension_table17	NA	NA
bcom	testbatch001	group003	18	DM01	runtype1dimension	dimension_table18	NA	NA
bcom	testbatch001	group003	19	DM01	runtype1dimension	dimension_table19	NA	NA
bcom	testbatch001	group004	20	DM01	runtype1dimension	dimension_table20	NA	NA
bcom	testbatch001	group004	21	DM01	runtype1dimension	dimension_table21	NA	NA
bcom	testbatch001	group004	22	DM01	runtype1dimension	dimension_table22	NA	NA
bcom	testbatch001	group004	23	DM01	runtype1dimension	dimension_table23	NA	NA
bcom	testbatch001	group005	24	DM01	runtype1dimension	dimension_table24	NA	NA
bcom	testbatch001	group005	25	DM01	runtype1dimension	dimension_table25	NA	NA
bcom	testbatch001	group006	26	DM01	runtype1dimension	dimension_table26	NA	NA
bcom	testbatch001	group006	27	DM01	runtype1dimension	dimension_table27	NA	NA
bcom	testbatch001	group007	28	DM01	runtype1dimension	dimension_table28	NA	NA
bcom	testbatch001	group008	29	DM01	runtype1dimension	dimension_table29	NA	NA
bcom	testbatch001	group008	30	DM01	runtype1dimension	dimension_table30	NA	NA
bcom	testbatch001	group008	31	DM01	runtype1dimension	dimension_table31	NA	NA
bcom	testbatch001	group008	32	DM01	runtype1dimension	dimension_table32	NA	NA
bcom	testbatch001	group008	33	DM01	runtype1dimension	dimension_table33	NA	NA
bcom	testbatch001	group008	34	DM01	runtype1dimension	dimension_table34	NA	NA
bcom	testbatch001	group008	35	DM01	runtype1dimension	dimension_table35	NA	NA
bcom	testbatch001	group008	36	DM01	runtype1dimension	dimension_table36	NA	NA
bcom	testbatch001	group008	37	DM01	runtype1dimension	dimension_table37	NA	NA
bcom	testbatch001	group008	38	DM01	runtype1dimension	dimension_table38	NA	NA
bcom	testbatch001	group008	39	DM01	runtype1dimension	dimension_table39	NA	NA
bcom	testbatch001	group008	40	DM01	runtype1dimension	dimension_table40	NA	NA
bcom	testbatch001	group008	41	DM01	runtype1dimension	dimension_table41	NA	NA
bcom	testbatch001	group008	42	DM01	runtype1dimension	dimension_table42	NA	NA
bcom	testbatch001	group008	43	DM01	runtype1dimension	dimension_table43	NA	NA
bcom	testbatch001	group008	44	DM01	runtype1dimension	dimension_table44	NA	NA
bcom	testbatch001	group008	45	DM01	runtype1dimension	dimension_table45	NA	NA
bcom	testbatch001	group008	46	DM01	runtype1dimension	dimension_table46	NA	NA
bcom	testbatch001	group008	47	DM01	runtype1dimension	dimension_table47	NA	NA
bcom	testbatch001	group008	48	DM01	runtype1dimension	dimension_table48	NA	NA

bcom	testbatch001	group008	49	DM01	runtype1dimension	dimension_table49	NA	NA
bcom	testbatch001	group008	50	DM01	runtype1dimension	dimension_table50	NA	NA
bcom	testbatch001	group008	51	DM01	runtype1dimension	dimension_table51	NA	NA
bcom	testbatch001	group008	52	DM01	runtype1dimension	dimension_table52	NA	NA
bcom	testbatch001	group008	53	DM01	runtype1dimension	dimension_table53	NA	NA
bcom	testbatch001	group008	54	DM01	runtype1dimension	dimension_table54	NA	NA
bcom	testbatch001	group008	55	DM01	runtype1dimension	dimension_table55	NA	NA
bcom	testbatch001	group008	56	DM01	runtype1dimension	dimension_table56	NA	NA
bcom	testbatch001	group008	57	DM01	runtype1dimension	dimension_table57	NA	NA
bcom	testbatch001	group008	58	DM01	runtype1dimension	dimension_table58	NA	NA
bcom	testbatch001	group008	59	DM01	runtype1dimension	dimension_table59	NA	NA
bcom	testbatch001	group008	60	DM01	runtype1dimension	dimension_table60	NA	NA
bcom	testbatch001	group008	61	DM01	runtype1dimension	dimension_table61	NA	NA
bcom	testbatch001	group008	62	DM01	runtype1dimension	dimension_table62	NA	NA
bcom	testbatch001	group008	63	DM01	runtype1dimension	dimension_table63	NA	NA
bcom	testbatch001	group008	64	DM01	runtype1dimension	dimension_table64	NA	NA
bcom	testbatch001	group008	65	DM01	runtype1dimension	dimension_table65	NA	NA
bcom	testbatch001	group008	66	DM01	runtype1dimension	dimension_table66	NA	NA
bcom	testbatch001	group008	67	DM01	runtype1dimension	dimension_table67	NA	NA
bcom	testbatch001	group008	68	DM01	runtype1dimension	dimension_table68	NA	NA
bcom	testbatch001	group008	69	DM01	runtype1dimension	dimension_table69	NA	NA
bcom	testbatch001	group008	70	DM01	runtype1dimension	dimension_table70	NA	NA
bcom	testbatch001	group008	71	DM01	runtype1dimension	dimension_table71	NA	NA
bcom	testbatch001	group008	72	DM01	runtype1dimension	dimension_table72	NA	NA
bcom	testbatch001	group008	73	DM01	runtype1dimension	dimension_table73	NA	NA
bcom	testbatch001	group008	74	DM01	runtype1dimension	dimension_table74	NA	NA
bcom	testbatch001	group008	75	DM01	runtype1dimension	dimension_table75	NA	NA
bcom	testbatch001	group008	76	DM01	runtype1dimension	dimension_table76	NA	NA
bcom	testbatch001	group008	77	DM01	runtype1dimension	dimension_table77	NA	NA
bcom	testbatch001	group008	78	DM01	runtype1dimension	dimension_table78	NA	NA
bcom	testbatch001	group008	79	DM01	runtype1dimension	dimension_table79	NA	NA
bcom	testbatch001	group008	80	DM01	runtype1dimension	dimension_table80	NA	NA
bcom	testbatch001	group008	81	DM01	runtype1dimension	dimension_table81	NA	NA
bcom	testbatch001	group008	82	DM01	runtype1dimension	dimension_table82	NA	NA
bcom	testbatch001	group008	83	DM01	runtype1dimension	dimension_table83	NA	NA
bcom	testbatch001	group008	84	DM01	runtype1dimension	dimension_table84	NA	NA

bcom	testbatch001	group008	85	DM01	runtype1dimension	dimension_table85	NA	NA
bcom	testbatch001	group008	86	DM01	runtype1dimension	dimension_table86	NA	NA
bcom	testbatch001	group008	87	DM01	runtype1dimension	dimension_table87	NA	NA
bcom	testbatch001	group008	88	DM01	runtype1dimension	dimension_table88	NA	NA
bcom	testbatch001	group008	89	DM01	runtype1dimension	dimension_table89	NA	NA
bcom	testbatch001	group008	90	DM01	runtype1dimension	dimension_table90	NA	NA
bcom	testbatch001	group008	91	DM01	runtype1dimension	dimension_table91	NA	NA
bcom	testbatch001	group008	92	DM01	runtype1dimension	dimension_table92	NA	NA
bcom	testbatch001	group008	93	DM01	runtype1dimension	dimension_table93	NA	NA
bcom	testbatch001	group008	94	DM01	runtype1dimension	dimension_table94	NA	NA
bcom	testbatch001	group008	95	DM01	runtype1dimension	dimension_table95	NA	NA
bcom	testbatch001	group008	96	DM01	runtype1dimension	dimension_table96	NA	NA
bcom	testbatch001	group008	97	DM01	runtype1dimension	dimension_table97	NA	NA
bcom	testbatch001	group008	98	DM01	runtype1dimension	dimension_table98	NA	NA
bcom	testbatch001	group008	99	DM01	runtype1dimension	dimension_table99	NA	NA
bcom	testbatch001	group008	100	DM01	runtype1dimension	dimension_table100	NA	NA
bcom	testbatch001	group008	101	DM01	runtype1dimension	dimension_table101	NA	NA
bcom	testbatch001	group008	102	DM01	runtype1dimension	dimension_table102	NA	NA
bcom	testbatch001	group999	103	DM01	rm	/seetl/seetldev/data/testbatch001.run.yes	NA	NA
bcom	testbatch001	group999	104	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/testbatch002.run.yes	NA	NA
bcom							NA	NA
bcom	testbatch002	group001	1	MMIO01	runmemorymappedio1	None	NA	NA
bcom	testbatch002	group002	2	MMIO01	runmemorymappedio2	None	NA	NA
bcom	testbatch002	group003	3	MMIO01	runmemorymappedio3	None	NA	NA
bcom	testbatch002	group004	4	MMIO01	runmemorymappedio4	None	NA	NA
bcom	testbatch002	group005	5	MMIO01	runmemorymappedio5	None	NA	NA
bcom	testbatch002	group006	6	MMIO01	runmemorymappedio6	None	NA	NA
bcom	testbatch002	group007	7	MMIO01	runmemorymappedio7	None	NA	NA
bcom	testbatch002	group008	8	MMIO01	runmemorymappedio8	None	NA	NA
bcom	testbatch002	group999	1	RM01	rm	/seetl/seetldev/data/testbatch002.run.yes	NA	NA
bcom							NA	NA
bcom	testbatch003	group001	1	CP01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/testbatch004.run.yes	NA	NA
bcom							NA	NA
bcom	testbatch004	group001	103	AS01	runasoc1	association_table_01	NA	NA

bcom	testbatch004	group002	104	AS01	runasoc1	association_table_02	NA	NA
bcom	testbatch004	group003	105	AS01	runasoc1	association_table_03	NA	NA
bcom	testbatch004	group004	107	AS01	runasoc1	association_table_04	NA	NA
bcom	testbatch004	group005	108	AS01	runasoc1	association_table_05	NA	NA
bcom	testbatch004	group006	109	AS01	runasoc1	association_table_06	NA	NA
bcom	testbatch004	group001	110	AS01	runasocload	association_table_01	NA	NA
bcom	testbatch004	group002	111	AS01	runasocload	association_table_02	NA	NA
bcom	testbatch004	group003	112	AS01	runasocload	association_table_03	NA	NA
bcom	testbatch004	group004	114	AS01	runasocload	association_table_04	NA	NA
bcom	testbatch004	group005	115	AS01	runasocload	association_table_05	NA	NA
bcom	testbatch004	group006	116	AS01	runasocload	association_table_06	NA	NA
bcom	testbatch004	group001	117	AS01	runasocupdate	association_table_01	NA	NA
bcom	testbatch004	group002	118	AS01	runasocupdate	association_table_02	NA	NA
bcom	testbatch004	group003	119	AS01	runasocupdate	association_table_03	NA	NA
bcom	testbatch004	group004	121	AS01	runasocupdate	association_table_04	NA	NA
bcom	testbatch004	group005	122	AS01	runasocupdate	association_table_05	NA	NA
bcom	testbatch004	group006	123	AS01	runasocupdate	association_table_06	NA	NA
bcom	testbatch004	group999	1	CP01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/testbatch005.run.yes	NA	NA
bcom	testbatch004	group999	2	RM01	rm	/seetl/seetldev/data/testbatch004.run.yes	NA	NA
bcom							NA	NA
bcom	testbatch005	group001	1	MMIO01	runmemorymappedio9	None	NA	NA
bcom	testbatch005	group002	2	MMIO01	runmemorymappedio10	None	NA	NA
bcom	testbatch005	group999	1	RM01	rm	/seetl/seetldev/data/testbatch005.run.yes	NA	NA
bcom	testbatch006	group001	1	CP01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/testbatch007.run.yes	NA	NA
bcom							NA	NA
bcom	testbatch007	group001	1	FA01	runfact1	fact_table_01	NA	NA
bcom	testbatch007	group001	2	FA01	runfact1	fact_table_02	NA	NA
bcom	testbatch007	group001	3	FA01	runfact1	fact_table_03	NA	NA
bcom	testbatch007	group001	4	FA01	runfact1	fact_table_04	NA	NA
bcom	testbatch007	group001	5	FA01	runfact1	fact_table_05	NA	NA
bcom	testbatch007	group001	6	FA01	runfact1	fact_table_06	NA	NA
bcom	testbatch007	group002	7	FA01	runfact1	fact_table_07	NA	NA
bcom	testbatch007	group002	8	FA01	runfact1	fact_table_08	NA	NA
bcom	testbatch007	group002	9	FA01	runfact1	fact_table_09	NA	NA

bcom	testbatch007	group002	10	FA01	runfact1	fact_table_10	NA	NA
bcom	testbatch007	group002	11	FA01	runfact1	fact_table_11	NA	NA
bcom	testbatch007	group002	12	FA01	runfact1	fact_table_12	NA	NA
bcom	testbatch007	group003	13	FA01	runfact1	fact_table_13	NA	NA
bcom	testbatch007	group003	14	FA01	runfact1	fact_table_14	NA	NA
bcom	testbatch007	group003	15	FA01	runfact1	fact_table_15	NA	NA
bcom	testbatch007	group003	16	FA01	runfact1	fact_table_16	NA	NA
bcom	testbatch007	group003	17	FA01	runfact1	fact_table_17	NA	NA
bcom	testbatch007	group003	18	FA01	runfact1	fact_table_18	NA	NA
bcom	testbatch007	group004	19	FA01	runfact1	fact_table_19	NA	NA
bcom	testbatch007	group004	20	FA01	runfact1	fact_table_20	NA	NA
bcom	testbatch007	group004	21	FA01	runfact1	fact_table_21	NA	NA
bcom	testbatch007	group004	22	FA01	runfact1	fact_table_22	NA	NA
bcom	testbatch007	group004	23	FA01	runfact1	fact_table_23	NA	NA
bcom	testbatch007	group004	24	FA01	runfact1	fact_table_24	NA	NA
bcom	testbatch007	group005	25	FA01	runfact1	fact_table_25	NA	NA
bcom	testbatch007	group005	26	FA01	runfact1	fact_table_26	NA	NA
bcom	testbatch007	group005	27	FA01	runfact1	fact_table_27	NA	NA
bcom	testbatch007	group005	28	FA01	runfact1	fact_table_28	NA	NA
bcom	testbatch007	group005	29	FA01	runfact1	fact_table_29	NA	NA
bcom	testbatch007	group005	30	FA01	runfact1	fact_table_30	NA	NA
bcom	testbatch007	group006	31	FA01	runfact1	fact_table_31	NA	NA
bcom	testbatch007	group006	32	FA01	runfact1	fact_table_32	NA	NA
bcom	testbatch007	group006	33	FA01	runfact1	fact_table_33	NA	NA
bcom	testbatch007	group006	34	FA01	runfact1	fact_table_34	NA	NA
bcom	testbatch007	group006	35	FA01	runfact1	fact_table_35	NA	NA
bcom	testbatch007	group006	36	FA01	runfact1	fact_table_36	NA	NA
bcom	testbatch007	group007	37	FA01	runfact1	fact_table_37	NA	NA
bcom	testbatch007	group007	38	FA01	runfact1	fact_table_38	NA	NA
bcom	testbatch007	group007	39	FA01	runfact1	fact_table_39	NA	NA
bcom	testbatch007	group007	40	FA01	runfact1	fact_table_40	NA	NA
bcom	testbatch007	group007	41	FA01	runfact1	fact_table_41	NA	NA
bcom	testbatch007	group007	42	FA01	runfact1	fact_table_42	NA	NA
bcom	testbatch007	group008	43	FA01	runfact1	fact_table_43	NA	NA
bcom	testbatch007	group008	44	FA01	runfact1	fact_table_44	NA	NA
bcom	testbatch007	group008	45	FA01	runfact1	fact_table_45	NA	NA

bcom	testbatch007	group008	46	FA01	runfact1	fact_table_46	NA	NA
bcom	testbatch007	group008	47	FA01	runfact1	fact_table_47	NA	NA
bcom	testbatch007	group008	48	FA01	runfact1	fact_table_48	NA	NA
bcom	testbatch007	group008	49	FA01	runfact1	fact_table_49	NA	NA
bcom	testbatch007	group008	50	FA01	runfact1	fact_table_50	NA	NA
bcom	testbatch007	group008	51	FA01	runfact1	fact_table_51	NA	NA
bcom	testbatch007	group008	52	FA01	runfact1	fact_table_52	NA	NA
bcom	testbatch007	group008	53	FA01	runfact1	fact_table_53	NA	NA
bcom	testbatch007	group008	54	FA01	runfact1	fact_table_54	NA	NA
bcom	testbatch007	group008	55	FA01	runfact1	fact_table_55	NA	NA
bcom	testbatch007	group008	56	FA01	runfact1	fact_table_56	NA	NA
bcom	testbatch007	group999	57	DM01	rm	/seetl/seetldev/data/testbatch007.run.yes	NA	NA
bcom	testbatch007	group999	58	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/testbatch008.run.yes	NA	NA
bcom	testbatch007	group999	59	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/memorymappedio1.kill.yes	NA	NA
bcom	testbatch007	group999	60	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/memorymappedio2.kill.yes	NA	NA
bcom	testbatch007	group999	61	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/memorymappedio3.kill.yes	NA	NA
bcom	testbatch007	group999	62	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/memorymappedio4.kill.yes	NA	NA
bcom	testbatch007	group999	63	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/memorymappedio5.kill.yes	NA	NA
bcom	testbatch007	group999	64	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/memorymappedio6.kill.yes	NA	NA
bcom	testbatch007	group999	65	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/memorymappedio7.kill.yes	NA	NA

bcom	testbatch007	group999	66	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/memorymappedio8.kill.yes	NA	NA
bcom	testbatch007	group999	67	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/memorymappedio9.kill.yes	NA	NA
bcom	testbatch007	group999	68	DM01	cp	/seetl/seetldev/data/asignalfile /seetl/seetldev/data/memorymappedio10.kill.yes	NA	NA
bcom	testbatch008	group001	1	FA01	runfactload1	fact_table_01	NA	NA
bcom	testbatch008	group001	2	FA01	runfactload1	fact_table_02	NA	NA
bcom	testbatch008	group001	3	FA01	runfactload1	fact_table_03	NA	NA
bcom	testbatch008	group001	4	FA01	runfactload1	fact_table_04	NA	NA
bcom	testbatch008	group001	5	FA01	runfactload1	fact_table_05	NA	NA
bcom	testbatch008	group001	6	FA01	runfactload1	fact_table_06	NA	NA
bcom	testbatch008	group002	7	FA01	runfactload1	fact_table_07	NA	NA
bcom	testbatch008	group002	8	FA01	runfactload1	fact_table_08	NA	NA
bcom	testbatch008	group002	9	FA01	runfactload1	fact_table_09	NA	NA
bcom	testbatch008	group002	10	FA01	runfactload1	fact_table_10	NA	NA
bcom	testbatch008	group002	11	FA01	runfactload1	fact_table_11	NA	NA
bcom	testbatch008	group002	12	FA01	runfactload1	fact_table_12	NA	NA
bcom	testbatch008	group003	13	FA01	runfactload1	fact_table_13	NA	NA
bcom	testbatch008	group003	14	FA01	runfactload1	fact_table_14	NA	NA
bcom	testbatch008	group003	15	FA01	runfactload1	fact_table_15	NA	NA
bcom	testbatch008	group003	16	FA01	runfactload1	fact_table_16	NA	NA
bcom	testbatch008	group003	17	FA01	runfactload1	fact_table_17	NA	NA
bcom	testbatch008	group003	18	FA01	runfactload1	fact_table_18	NA	NA
bcom	testbatch008	group004	19	FA01	runfactload1	fact_table_19	NA	NA
bcom	testbatch008	group004	20	FA01	runfactload1	fact_table_20	NA	NA
bcom	testbatch008	group004	21	FA01	runfactload1	fact_table_21	NA	NA
bcom	testbatch008	group004	22	FA01	runfactload1	fact_table_22	NA	NA
bcom	testbatch008	group004	23	FA01	runfactload1	fact_table_23	NA	NA
bcom	testbatch008	group004	24	FA01	runfactload1	fact_table_24	NA	NA
bcom	testbatch008	group005	25	FA01	runfactload1	fact_table_25	NA	NA
bcom	testbatch008	group005	26	FA01	runfactload1	fact_table_26	NA	NA
bcom	testbatch008	group005	27	FA01	runfactload1	fact_table_27	NA	NA

bcom	testbatch008	group005	28	FA01	runfactload1	fact_table_28	NA	NA
bcom	testbatch008	group005	29	FA01	runfactload1	fact_table_29	NA	NA
bcom	testbatch008	group005	30	FA01	runfactload1	fact_table_30	NA	NA
bcom	testbatch008	group006	31	FA01	runfactload1	fact_table_31	NA	NA
bcom	testbatch008	group006	32	FA01	runfactload1	fact_table_32	NA	NA
bcom	testbatch008	group006	33	FA01	runfactload1	fact_table_33	NA	NA
bcom	testbatch008	group006	34	FA01	runfactload1	fact_table_34	NA	NA
bcom	testbatch008	group006	35	FA01	runfactload1	fact_table_35	NA	NA
bcom	testbatch008	group006	36	FA01	runfactload1	fact_table_36	NA	NA
bcom	testbatch008	group007	37	FA01	runfactload1	fact_table_37	NA	NA
bcom	testbatch008	group007	38	FA01	runfactload1	fact_table_38	NA	NA
bcom	testbatch008	group007	39	FA01	runfactload1	fact_table_39	NA	NA
bcom	testbatch008	group007	40	FA01	runfactload1	fact_table_40	NA	NA
bcom	testbatch008	group007	41	FA01	runfactload1	fact_table_41	NA	NA
bcom	testbatch008	group007	42	FA01	runfactload1	fact_table_42	NA	NA
bcom	testbatch008	group008	43	FA01	runfactload1	fact_table_43	NA	NA
bcom	testbatch008	group008	44	FA01	runfactload1	fact_table_44	NA	NA
bcom	testbatch008	group008	45	FA01	runfactload1	fact_table_45	NA	NA
bcom	testbatch008	group008	46	FA01	runfactload1	fact_table_46	NA	NA
bcom	testbatch008	group008	47	FA01	runfactload1	fact_table_47	NA	NA
bcom	testbatch008	group008	48	FA01	runfactload1	fact_table_48	NA	NA
bcom	testbatch008	group008	49	FA01	runfactload1	fact_table_49	NA	NA
bcom	testbatch008	group008	50	FA01	runfactload1	fact_table_50	NA	NA
bcom	testbatch008	group008	51	FA01	runfactload1	fact_table_51	NA	NA
bcom	testbatch008	group008	52	FA01	runfactload1	fact_table_52	NA	NA
bcom	testbatch008	group008	53	FA01	runfactload1	fact_table_53	NA	NA
bcom	testbatch008	group008	54	FA01	runfactload1	fact_table_54	NA	NA
bcom	testbatch008	group008	55	FA01	runfactload1	fact_table_55	NA	NA
bcom	testbatch008	group008	56	FA01	runfactload1	fact_table_56	NA	NA
bcom	testbatch008	group999	57	DM01	rm	/seetl/seetldev/data/testbatch008.run.yes	NA	NA

Process Commands

In previous releases of SeETL[RT] the commands to execute the command were stored in command files that were visible to the path. This method of calling SeETL[RT] commands is still supported. However in 2.1 we have added the ability to place the process commands into the SeETL[DT] Spreadsheet. SeETL[DT] will then move the commands to the process commands table. The Batch Scheduler will read the process commands table to determine if a command in the commands table is in the process commands table. If it is, the Batch Scheduler will substitute the process command prior to then issuing the command to the operating system.

Row Type	PK Integer Key	Process Name	Process Command
pcom	1	testcommand	Echo ?1 more ?2 more ?3
pcom	2	runfact1	CTLAT02 DBConnectionInParameter=DSN=SEETL3000 InCatalogName=SEETL3000 InSchemaName=DEVSTAGE InTableName=?1 DBConnectionOutParameter=DSN=SEETL3000 OutCatalogName=IDWDEV_ORA OutSchemaName=IDWDEV OutTableName=?1_INS ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes CTLF007FileName=/seetl/seetldev/data/cdwf007_?1.dat CTLF001FileName=/seetl/seetldev/data/cdwf007_?1.dat SourceDataBase=Oracle9 TargetDataBase=Oracle9 TranslateNewline=Yes TranslateNewlineTo=+ NumberRowsToTransfer=100000 MemoryMappedIODirectory=/seetl/mmio/
pcom	3	runfactload1	CTLU001 DBConnectionOutParameter=DSN=IDWDEV_ORA OutSchemaName=IDWDEV OutTableName=?1_INS ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes WorkFileName=/seetl/seetldev/data/cdwf007_?1.dat InsertUpdateOption=InsertTheUpdate DataMovementOption=Load LoadFactTable=Yes TargetDataBase=Oracle9 AutoCommit=No CommitFrequency=10000 InputTableOrFileCanBeEmpty=Yes
pcom	4	runasoc1	CTLAS01 DBConnectionInParameter=DSN=IDW_DEVSTAGE_ORA InCatalogName=IDW_DEVSTAGE_ORA InSchemaName=DEVSTAGE InTableName=?1 DBConnectionOutParameter=DSN=IDWDEV_ORA OutCatalogName=IDWDEV_ORA OutSchemaName=IDWDEV OutTableName=?1 ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes CTLF007FileName=/seetl/seetldev/data/cdwf007_?1.dat CTLF008FileName=/seetl/seetldev/data/cdwf008_?1.dat SourceDataBase=Oracle9 TargetDataBase=Oracle9 MemoryMappedIODirectory=/seetl/mmio/
pcom	5	runasocload	CTLU001 DBConnectionOutParameter=DSN=IDWDEV_ORA OutCatalogName=IDWDEV_ORA OutSchemaName=IDWDEV OutTableName=?1_INS ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes WorkFileName=/seetl/seetldev/data/cdwf005_?1.dat InsertUpdateOption=InsertOnly DataMovementOption=Load LoadFactTable=No TargetDataBase=Oracle9 AutoCommit=No CommitFrequency=10000 InputTableOrFileCanBeEmpty=Yes
pcom	6	runasocupdate	CTLU001 DBConnectionOutParameter=DSN=IDWDEV_ORA OutCatalogName=IDWDEV OutSchemaName=IDWDEV OutTableName=?1_UPD ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes WorkFileName=/seetl/seetldev/data/cdwf006_?1.dat InsertUpdateOption=UpdateThenInsert DataMovementOption=Load LoadFactTable=No SourceDataBase=Oracle9 TargetDataBase=Oracle9

			AutoCommit=No CommitFrequency=10000 InputTableOrFileCanBeEmpty=Yes
pcom	7	runtype1dimension	CTLDM01 DBConnectionInParameter=DSN=IDW_DEVSTAGE_ORA InCatalogName=IDW_DEVSTAGE_ORA InSchemaName=DEVSTAGE InTableName=?1 DBConnectionOutParameter=DSN=IDWDEV_ORA OutCatalogName=IDWDEV_ORA OutSchemaName=IDWDEV OutTableName=?1 ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes AutoCommit=No CommitFrequency=10000 SourceDatabase=Oracle9 TargetDatabase=Oracle9
pcom	8	runmemorymappedio1	CTLU012 DBConnectionInParameter=DSN=IDWDEV_ORA InSchemaName=IDWDEV InTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes SourceDataBase=Oracle9 TargetDataBase=Oracle9 WaitFileName=/seetl/seetldev/data/memorymappedio1.wait.yes KillFileName=/seetl/seetldev/data/memorymappedio1.kill.yes LoadDimensionTableGroup=GROUP1 MemoryMappedIODirectory=/seetl/mmio/
pcom	9	runmemorymappedio2	CTLU012 DBConnectionInParameter=DSN=IDWDEV_ORA InSchemaName=IDWDEV InTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes SourceDataBase=Oracle9 TargetDataBase=Oracle9 WaitFileName=/seetl/seetldev/data/memorymappedio2.wait.yes KillFileName=/seetl/seetldev/data/memorymappedio2.kill.yes LoadDimensionTableGroup=GROUP2 MemoryMappedIODirectory=/seetl/mmio/
pcom	10	runmemorymappedio3	CTLU012 DBConnectionInParameter=DSN=IDWDEV_ORA InSchemaName=IDWDEV InTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes SourceDataBase=Oracle9 TargetDataBase=Oracle9 WaitFileName=/seetl/seetldev/data/memorymappedio3.wait.yes KillFileName=/seetl/seetldev/data/memorymappedio3.kill.yes LoadDimensionTableGroup=GROUP3 MemoryMappedIODirectory=/seetl/mmio/
pcom	11	runmemorymappedio4	CTLU012 DBConnectionInParameter=DSN=IDWDEV_ORA InSchemaName=IDWDEV InTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes SourceDataBase=Oracle9 TargetDataBase=Oracle9 WaitFileName=/seetl/seetldev/data/memorymappedio4.wait.yes KillFileName=/seetl/seetldev/data/memorymappedio4.kill.yes LoadDimensionTableGroup=GROUP4 MemoryMappedIODirectory=/seetl/mmio/
pcom	12	runmemorymappedio5	CTLU012 DBConnectionInParameter=DSN=IDWDEV_ORA InSchemaName=IDWDEV InTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes SourceDataBase=Oracle9 TargetDataBase=Oracle9 WaitFileName=/seetl/seetldev/data/memorymappedio5.wait.yes KillFileName=/seetl/seetldev/data/memorymappedio5.kill.yes LoadDimensionTableGroup=GROUP5 MemoryMappedIODirectory=/seetl/mmio/
pcom	13	runmemorymappedio6	CTLU012 DBConnectionInParameter=DSN=IDWDEV_ORA InSchemaName=IDWDEV InTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes SourceDataBase=Oracle9 TargetDataBase=Oracle9 WaitFileName=/seetl/seetldev/data/memorymappedio6.wait.yes KillFileName=/seetl/seetldev/data/memorymappedio6.kill.yes LoadDimensionTableGroup=GROUP6 MemoryMappedIODirectory=/seetl/mmio/
pcom	14	runmemorymappedio7	CTLU012 DBConnectionInParameter=DSN=IDWDEV_ORA InSchemaName=IDWDEV InTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes SourceDataBase=Oracle9 TargetDataBase=Oracle9 WaitFileName=/seetl/seetldev/data/memorymappedio7.wait.yes KillFileName=/seetl/seetldev/data/memorymappedio7.kill.yes LoadDimensionTableGroup=GROUP7 MemoryMappedIODirectory=/seetl/mmio/

pcom	15	runmemorymappedio8	CTLU012 DBConnectionInParameter=DSN=IDWDEV_ORA InSchemaName=IDWDEV InTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes SourceDataBase=Oracle9 TargetDataBase=Oracle9 WaitFileName=/seetl/seetldev/data/memorymappedio8.wait.yes KillFileName=/seetl/seetldev/data/memorymappedio8.kill.yes LoadDimensionTableGroup=GROUP8 MemoryMappedIODirectory=/seetl/mmio/
pcom	16	runmemorymappedio9	CTLU012 DBConnectionInParameter=DSN=IDWDEV_ORA InSchemaName=IDWDEV InTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes SourceDataBase=Oracle9 TargetDataBase=Oracle9 WaitFileName=/seetl/seetldev/data/memorymappedio9.wait.yes KillFileName=/seetl/seetldev/data/memorymappedio9.kill.yes LoadDimensionTableGroup=GROUP9 MemoryMappedIODirectory=/seetl/mmio/
pcom	17	runmemorymappedio10	CTLU012 DBConnectionInParameter=DSN=IDWDEV_ORA InSchemaName=IDWDEV InTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes SourceDataBase=Oracle9 TargetDataBase=Oracle9 WaitFileName=/seetl/seetldev/data/memorymappedio10.wait.yes KillFileName=/seetl/seetldev/data/memorymappedio10.kill.yes LoadDimensionTableGroup=GROUP10 MemoryMappedIODirectory=/seetl/mmio/
pcom	18	runddl	CTLU002 DBConnectionOutParameter=DSN=IDWDEV_ORA OutCatalogName=IDWDEV_ORA OutSchemaName=IDWDEV OutTableName=not_used ErrorMessageOutput=TargetDatabase DebugLevel=0 Audit=Yes TargetDatabase=Oracle9 SQLFileName=/seetl/seetldev/ddl/?1.ddl

This page is intentionally left blank.