# Cruise Object Conversion Utility Requirements

Contents

[Cruise Object Conversion Utility Requirements 1](#_Toc324251028)

[References 1](#_Toc324251029)

[Data typing 2](#_Toc324251030)

[Table constraints 2](#_Toc324251031)

[Data integrity and data audit mechanisms 2](#_Toc324251032)

[Methods for converting data 2](#_Toc324251033)

[Cruise Table 2](#_Toc324251034)

[Cutting Unit Table 3](#_Toc324251035)

[Stratum Table 3](#_Toc324251036)

[SubPopulation Table 4](#_Toc324251037)

[Count Table 5](#_Toc324251038)

[Plot Table 5](#_Toc324251039)

[Tree table 5](#_Toc324251040)

[Log Table 6](#_Toc324251041)

[Volume Equation Table 7](#_Toc324251042)

[Global Config Table 7](#_Toc324251043)

[Message Log Table 8](#_Toc324251044)

[Weight Equation Table 8](#_Toc324251045)

[Value Equation Table 9](#_Toc324251046)

[QA Equation Table 9](#_Toc324251047)

### References

In the metakit cruise file, tables are interconnected via a set of key fields: for example entries in the metakit log table reference an entry in the tree table using a combination of cutting unit code, plot number, and tree number. In SQL tables are interconnected using an integer foreign key value that corresponds to a primary key value in the referenced table; therefore in the new schema, the log table will no longer need fields to store the cutting unit code, plot number, and tree number to reference a single tree entry.

### Data typing

Data in the SQLite database is loosely typed, meaning that a column that has a declared type of integer, can actually hold data of any type (Booleans, floating point numbers, strings…) without throwing any error. However the library that has been designed to read the database file is strongly typed and will throw an error if it reads data that is not the same type that is expected. There are some instances where the data type in the metakit file is not the same as the data type in the new SQL schema and a conversion must be done.

### Table constraints

To help prevent accidental data corruption, constraints have been defined to enforce required fields to always be filled in, non-repetition of unique data, and the continuity of relationships between tables. If data given to the database violates any constraints, an error will be thrown and no changes will be made to the database.

### Data integrity and data audit mechanisms

In the new SLQ schema each table in the database has a CreatedBy, CreatedDate, ModifiedBy, and ModifiedDate column that allows the lifecycle of all the data in the database to be tracked. The user of the database does not need to provide values for CreatedDate and ModifiedDate, because they are automaticly handled by triggers in the database, however, CreatedBy is required whenever data is inserted, and ModifiedBy should be provided when updating an entry.

If the CruiseDAL library is used for entering data into the database, the user will not need to worry about setting values for CreatedBy or ModifiedBy, because the CruiseDAL will use the value from its User property to fill in the values, on saving objects to the database.

## Methods for converting data

Below, all the methods to convert a single entry or group of entries from the metakit cruise file to the SQLite cruise file, listed by the table in the metakit that the data is being read from.

### Cruise Table

Preconditions:

The metakit file should only contain one entry for the cruise table.

Postconditions:

One entry created in the sale table of the database file, with a Sale\_CN value set to 1, and Calendar Year converted from a string to an integer.

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| SaleNo | Sale number | SaleNumber |
| SaleName | Sale name | Name |
| Region | Region number | Region |
| Forest | Forest number | Forest |
| District | District number | District |
| MeasSys | Measurement system | Measurment |
| CalYear | Calendar year | Calendaryear |
| Purpose | Purpose of cruise | Purpose |
| Remarks | Remarks | Remarks |

### Cutting Unit Table

Preconditions:

Metakit CuttingUnitTable contains at least one entry

Post conditions:

For every entry in CuttingUnitTable one entry is created in the database CuttingUnit table

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| CutUnit | Cutting unit code | Code |
| Area | Cutting unit area | Area |
| Description | Cutting unit description | Description |
| LogMeth | Logging method | LoggingMethod |
| PayUnit | Payment unit code | PaymentUnit |

### Stratum Table

Preconditions:

StratumTable contains at least one entry and all entries from the CuttingUnitTable have been read, so references can be built between the two tables

Postconditions:

For each entry in StratumTable one entry is created in the database Stratum table, and for each comma separated string value in the Units field, one entry is created in the database CuttingUnitStratum mapping table, thereby, linking each StratumTable entry to one or more cutting unit entries in the database.

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| Stratum | Stratum code | Code |
| Description | Stratum description | Description |
| CruzMeth | Cruise method | Method |
| BAF | Basal area factor | BasalAreaFactor |
| FPSize | Fixed plot size | FixedPlotSize |
| Units | List of cutting units | N/A |
| Month | strata month | Month |
| Year | Strata year | Year |

### SubPopulation Table

Preconditions: SubPopulation table contains at least one entry, and stratum table has been read.

Post conditions:

For each unique combination of Sample group code and stratum code in SubPopulation, one entry has been created in the database SampleGroup table.

For each unique combination of all the fields corresponding to a TreeDefaultValue (see below), one entry has been created in the TreeDefaultValue table.

For each entry in the subpopulation table, one entry has been created in the SampleGroupTreeDefaultValue table, so that an entry in SampleGroup table corresponding to that subpopulation’s stratum code and sample group code is linked to an entry in the TreeDefaultValue table corresponding to the subpopulation’s Tree default data

Additional postcondition: for each entry created in the TreeDefaultValue table, the values for 'CDefP', 'HDefP', 'CDefS', 'HDefS', 'RecDefP' must be converted to float values

Tree Default Value fields:

|  |  |
| --- | --- |
| Field Name in metakit file | Coresponding field name in SQLite database |
| ProdP | PrimaryProduct |
| Sp | Species |
| LD | LiveDead |
| CDefP | CullPrimary |
| HDefP | HiddenPrimary |
| CDefS | CullSecondary |
| HDefS | HiddenSecondary |
| RedDefP | Recoverable |
| ContrSpec | ContractSpecies |
| TreeGrade | TreeGrade |
| MrchHtLL | MerchHeightLogLength |
| MrchHtT | MerchHeightType |
| FC | FormClass |
| DBHBTR | BarkThicknessRatio |
| AvgZForm | AverageZ |

Sample Group fields

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| SG | Sample group | Code |
| UOMP | Unit of measure primary product | UOM |
| ProdP | Primary product | PrimaryProduct |
| ProdS | Secondary product | SecondaryProduct |
| LD | Live or dead code | DefaultLiveDead |
| Freq | Frequency | SamplingFrequency |
| KZ | KZ value | KZ |
| Description | Description | Description |
| key | Tally hot key | TallyHotKey |
| YC | Yield component | YieldComponent |

### Count Table

Preconditions : CuttingUnitTable, and SubPopulation have been read

Postcondition: For each entry in the count table one entry has been created in the CountTree table

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| Sp | Species code | Species |
| Count | Tally count | TreeCount |
| SumKPI | Sum KPI | SumKPI |

### Plot Table

Preconditions: CuttingUnitTable and StratumTable have been read

Post conditions: For each entry in the plot table one entry has been created in the database Plot table and values for XCoord, YCoord, and ZCoord have been converted to float values

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| Plot | Plot number | PlotNumber |
| SlopePct | Plot slope percent | Slope |
| Aspect | Plot Aspect | Aspect |
| NullPlot | Null plot flag | IsEmpty |
| XCoord | Plot X coordinate | XCoordinate |
| YCoord | Plot Y coordinate | YCoordinate |
| ZCoord | Plot Z coordinate | Zcoordinate |
| MetaData | MetaData | MetaData |
| Remark | Remark | Remarks |

### Tree table

Preconditions: CuttingUnitTable, StratumTable, SubPopulation and PlotTable have been read.

Postconditions: For each entry in TreeTable one entry has been created in the database Tree table and the value for crown ratio has been converted to a float type

Special conditions:

* If the species is not defined for the tree and the tree number is 0 add the KPI value from that tree to the KPI value for that tree’s plot, and don’t create a tree entry.
* If the cruise method for the tree’s stratum is S3P or 3P add the value from the tree’s KPI to the sumKPI in the the Tree’s corresponding CountTree entry, and if the Count Measure (CM) value for the tree is ‘C’ do not create an entry in the tree table for that tree.
* (removed condition) Otherwise if the tree has a 3Pcode == 3 add the tree to a sample group called 3P-<pp> where pp is the primary product
* If the cruise method the tree’s stratum is STR or 3P and the tree is going to be added to the tree table, increment the TreeCount value in the Count Tree table for the corresponding entry.
* (removed) If the plot number is unable to resolve to an entry in the plot table and the cruise method, provided by the stratum of that tree, is STR, 3P, 100, S3P, or 3 P, a new entry has been created in the plot table and is referenced by the current tree and the user has been requested to fill in data for newly created plot.
* If the secondary product for the sample group of a tree is undefined, set the secondary product of the sample group to that of the tree. If the secondary product of the sample group is defined but different from the tree’s then (… ?)

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| Tree | Tree number | TreeNumber |
| Sp | Species code | Species |
| CM | Count or measure code | CountOrMeasure |
| TreeCnt | Tree count | Count |
| KPI | KPI estimate | KPI |
| SDefP | % Seen defect primary product | SeenDefectPrimary |
| SDefS | % Seen defect secondary product | SeenDefectSecondary |
| RecDefP | % Recoverable primary product | RecoverablePrimary |
| Initials | Marker's initials | Initials |
| LD | Live or dead code | LiveDead |
| TreeGrade | Tree grade | Grade |
| HgtFLL | Height to first live limb | HeightToFirstLiveLimb |
| PoleLen | Pole length | PoleLength |
| Clear | Clear face | ClearFace |
| CR | Crown ratio | CrownRatio |
| DBH | DBH outside bark | DBH |
| DRCOB | Diameter at root collar outside bark | DRC |
| TotHt | Total height | TotalHeight |
| MrchHtP | Merchantable height primary product | MerchHeightPrimary |
| MrchHtS | Merchantable height secondary product | MerchHeightSecondary |
| FC | Form class code | FormClass |
| UStemDOB | Upper stem diameter outside bark | UpperStemDOB |
| HgtUStem | Height of upper stem measurement | UpperStemHeight |
| DBHDBT | DBH double bark thickness | DBHDoubleBarkThickness |
| TopDIBP | Top DIB primary product | TopDIBPrimary |
| TopDIBS | Top DIB secondary product | TopDIBSecondary |
| DefectCode | Defect code | DefectCode |
| DiamDefPt | Diameter at defect point | DiameterAtDefect |
| VoidPct | Void percent | VoidPercent |
| Remarks | Tree remarks | Remarks |

### Log Table

Preconditions: CuttingUnitTable, StratumTable, SubPopulation, PlotTable and TreeTable have been read

Postconditions: For each entry in the log table an entry has been created in the database log table

|  |  |  |  |
| --- | --- | --- | --- |
| Metakit field Name | | Description | Corresponding SQLite Field Name |
| LogNum | Log number | | LogNumber |
| Grade | Log grade | | Grade |
| PctSeenDef | % Seen defect | | SeenDefect |
| PctRecChip | % Recoverable chips | | PercentRecoverable |
| SErD | Small end rounded diameter | | SmallEndDiameter |
| LErD | Large end rounded diamter | | LargeEndDiameter |
| DBT | Diameter bark thickness | | BarkThickness |
| Length | Log length | | Length |
| GBDFT | Gross board foot volume | | GrossBoardFoot |
| GCUFT | Gross cubic foot volume | | GrossCubicFoot |
| NBDFT | Net board foot volume | | NetBoardFoot |
| NCUFT | Net cubic foot volume | | NetCubicFoot |
| BDFTRem | BDFT removed volume | | BoardFootRemoved |
| CUFTRem | CUFT removed volume | | CubicFootRemoved |
| DIBCls | Log DIB class | | DIBClass |
| Sort | Log sort (export grade) | | ExportGrade |

### Volume Equation Table

Preconditions: None

Postconditions: For each entry in the volume equation table one entry has been created in the database VolumeEquation table, and CalcTotal, CalcBoard, CalcCubic, CalcCord, and CalcTopWood have been assigned values extracted from VolEqFlags, such that for bit-string, assigned to VolEqFlags, A that is 5 characters long or more, CalcTotal = A[0], CalcBoard = A[1], CalcCubic = A[2], CalcCords = A[3], CalcTopwood = A[4]

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| VolEqSpecie | Volume equation specie | Species |
| VolEqPP | Volume equation primary product | PrimaryProduct |
| VolEqNum | Volume equation number | VolumeEquationNumber |
| VolEqStump | Volume equation stump height | StumpHeight |
| VolEqTopDIBP | Volume equation min top DIB primary | TopDIBPrimary |
| VolEqTopDIBS | Volume equation min top dib secondary | TopDIBSecondary |
| ValEqSpecie | Value equation specie | Species |
| ValEqPP | Value equation primary product | PrimaryProduct |
| ValEqNum | Value equation number | VolumeEquationNumber |

### Global Config Table

Preconditions: None

Postconditions: For each entry in the GlobalConfigTable, if the value of Key is “Reports” then one entry has been created in the database Reports table, otherwise one entry has been created in the database Globals table.

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| Block | Block value | Block |
| Key | Key value | Key |
| Value | Value | Value |

### Message Log Table

Preconditions: None

Post conditions: For each entry in MessageLog one entry has been created in the database MessageLog table

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| Program | Program | Program |
| Msg | Message | Message |
| Date | Date | Date |
| Time | Time | Time |
| Level | Level | Level |

### Weight Equation Table

Preconditions: None

Postconditions: For each entry in WeightEquationTable one entry has been created in the database WeightEquation table and one entry has been created in the database BiomassEquation table for each biomass component.

Additional notes for reading biomass data:

There are 12 fields in the weight equation, each having a prefix of BioEq, BioMoist, or BioRemv - that correspond, respectively to the Equation, PercentMoisture, and PercentRemoved fields in the BiomassEquation table– and a suffix of Fol, LBr, DBr, or TotTree corresponding to a biomass component. If the fields for a given component are non-zero, then one entry must be created in the database BiomassEquation table and populated with data for that component, along with the species, and BioPr (Product) values from the weight table.

|  |  |  |
| --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |
| WgtEqSpecie | Weight equation specie | Species |
| WgtEqPP | Weight equation primary product | Product |
| WgtEqFactor | Weight equation weight factor primary | WeightFactorPrimary |
| WgtEqPercent | Weight equation percent removed primary | PercentRemovedPrimary |
| WgtEqLiveDead | Weight Live Dead code | LiveDead |
| WgtEqFactor2 | Weight equation weight factor secondary | WeightFactorSecondary |
| WgtEqPercent2 | Weight equation percent removed secondary | PercentRemovedSecondary |
| WgtEqSP | Weight equation secondary product code | SecondaryProduct |

### Value Equation Table

Preconditions: None

Postconditions: for each entry in ValueEquationTable one entry has been created in the ValueEquation

### QA Equation Table

Preconditions: None

Postconditions: for each entry in QAEquationTable one entry has been created in the QualityAdjEquation table

|  |  |  |  |
| --- | --- | --- | --- |
| Metakit field Name | Description | Corresponding SQLite Field Name |  |
| QaEqSpecie | QA equation species | Species |  |
| QaEqNum | QA equation number | QualityAdjEq |  |
| QaEqYear | QA equation pricing month/year | Year |  |
| QaEqCoef1 | QA equation coefficient 1 | Coefficient1 |  |
| QaEqCoef2 | QA equation coefficient 2 | Coefficient2 |  |
| QaEqCoef3 | QA equation coefficient 3 | Coefficient3 |  |
| QaEqCoef4 | QA equation coefficient 4 | Coefficient4 |  |
| QaEqCoef5 | QA equation coefficient 5 | Coefficient5 |  |
| QaEqCoef6 | QA equation coefficient 6 | Coefficient6 |  |