

ICS Konfigurationshandbuch

Zusammenfassung

Diese Dokumentation beschreibt die Konfiguration des infoGrips Conversion System (ICS).

Die Dokumentation darf nur mit Erlaubnis der infoGrips GmbH vervielfältigt werden.

Inhaltsverzeichnis

1. Allgemeiner Aufbau von ICS Konfigurationen	13
1.1. infoGrips Conversion System (ICS)	13
1.2. Grundstruktur von ICS Konfigurationen	14
1.3. Inhalt der .cfg Datei	15
1.4. Der RUN1 Algorithmus	18
1.4.1. Abfrage von Benutzerparametern	18
1.4.2. Inputquellen	19
1.4.3. Die INOUT Map	20
1.4.4. Ausführen der Abbildungsvorschrift	21
1.4.5. Die MACRO Map	21
1.4.6. Benutzerprozeduren in Abbildungsvorschriften	22
1.4.7. Triggerprozeduren	23
2. Beispielkonfigurationen	25
2.1. Einleitung	25
2.2. Konfigurationen mit einem Inputmodul	25
2.2.1. Anzeigekonfiguration	25
2.2.2. Statistikkonfiguration	27
2.2.3. Checkerkonfiguration	29
2.3. Konfiguration mit Input- und Outputmodul	31
2.4. Konfiguration mit Verarbeitungsmodul	33
3. Konfigurieren mit ICS	37
3.1. Einleitung	37
3.2. Verwendung von Editoren	37
3.3. Ausführen der Konfigurationen	38
3.4. Hinweise und Tips	39
3.4.1. Fehlersuche in ICS Konfigurationen (Debugging)	39
3.4.2. Batchaufruf von ICS Konfigurationen	40
A. Input Module	41
1. Einleitung	41
2. Modul ARCGISIN - ESRI Geodatabase lesen	41
2.1. Allgemeines	41
2.2. ESRI Lizenz	41
2.3. Parametermap ARCGISIN_PARAM	41
2.4. ArcGIS SDE Connect	42
2.5. Objektmodell	43
2.6. Exportierte Prozeduren und Methoden	44
2.7. Skriptbeispiel	44
2.8. Bestehende Konfigurationen IL2GDB/IL2SDE oder GDB2IL/SDE2IL nach IL2ARCGIS/ARCGIS2IL migrieren	45
3. Modul DBIN - ODBC-Datenbank lesen	45
3.1. Allgemeines	45
3.2. Abhängigkeiten von anderen Modulen	45
3.3. Parametermap DB_PARAM	45
3.4. Parametermap DBIN_PARAM	46
3.5. Objektmodell	46
3.6. Datenbankmodell	46
3.7. Exportierte Prozeduren und Methoden	47
3.8. Skriptbeispiel	48
4. Modul DGNIN - Bentley Microstation DGN lesen	49
4.1. Allgemeines	49
4.2. Abhängigkeiten von anderen Modulen	49
4.3. Parametermap DGNIN_PARAM	49
4.4. Map für Textjustierung	49
4.5. Objektmodell	50

4.6. Exportierte Prozeduren und Methoden	51
4.7. Skriptbeispiel	51
5. Modul DXFIN - AutoCAD DXF lesen	52
5.1. Allgemeines	52
5.2. Abhängigkeiten von anderen Modulen	52
5.3. Parametermap DXFIN_PARAM	52
5.4. Objektmodell	53
5.5. Exportierte Prozeduren und Methoden	56
5.6. Skriptbeispiel	56
6. Modul FILEGDBIN - ESRI File-Geodatabase lesen	57
6.1. Allgemeines	57
6.2. ESRI Lizenz	57
6.3. Parametermap FILEGDBIN_PARAM	57
6.4. Objektmodell	57
6.5. Exportierte Prozeduren und Methoden	58
6.6. Skriptbeispiel	58
7. Modul GEOJSONIN - GeoJSON lesen	59
7.1. Allgemeines	59
7.2. Abhängigkeiten von anderen Modulen	59
7.3. Parametermap GEOJSONIN_PARAM	59
7.4. Objektmodell	59
7.5. Exportierte Prozeduren und Methoden	60
7.6. Skriptbeispiel	60
8. Modul GMMDBIN - Intergraph GeoMedia ACCESS Datenbank lesen	61
8.1. Allgemeines	61
8.2. Abhängigkeiten von anderen Modulen	61
8.3. Parametermap DB_PARAM	61
8.4. Parametermap DBIN_PARAM	62
8.5. Parametermap GMMDBIN_PARAM	62
8.6. Objektmodell	62
8.7. Exportierte Prozeduren und Methoden	62
8.8. Skriptbeispiel	64
9. Modul GMORAIN - Intergraph GeoMedia Oracle Datenbank lesen	65
9.1. Allgemeines	65
9.2. Abhängigkeiten von anderen Modulen	65
9.3. Parametermap DB_PARAM	65
9.4. Parametermap DBIN_PARAM	66
9.5. Parametermap ORAIN_PARAM	66
9.6. Parametermap GMORAIN_PARAM	66
9.7. Objektmodell	66
9.8. Spezielles	67
9.9. Exportierte Prozeduren und Methoden	67
9.10. Skriptbeispiel	69
10. Modul IFCIN - Industry Foundation Classes IFC lesen	70
10.1. Allgemeines	70
10.2. Parametermap IFCIN_PARAM	70
10.3. Objektmodell	70
10.4. Exportierte Prozeduren und Methoden	71
10.5. Skriptbeispiel	72
11. Modul IL2IN - INTERLIS 2 lesen	72
11.1. Allgemeines	72
11.2. Abhängigkeiten von anderen Modulen	73
11.3. Parametermap IL2IN_PARAM	73
11.4. Objektmodell	74
11.5. Exportierte Prozeduren und Methoden	74
11.6. Skriptbeispiel	75
12. Modul ILIN - INTERLIS 1 lesen	76
12.1. Allgemeines	76

12.2. Abhängigkeiten von anderen Modulen	76
12.3. Parametermap ILIN_PARAM	76
12.4. Objektmodell	77
12.5. Exportierte Prozeduren und Methoden	79
12.6. Skriptbeispiel	80
13. Modul ILTOPO - INTERLIS 1 lesen mit Topologieberechnung	81
13.1. Allgemeines	81
13.2. Abhängigkeiten von anderen Modulen	81
13.3. Parametermap ILIN_PARAM	81
13.4. Parametermap ILIN_TOPO	82
13.5. Objektmodell	83
13.6. Exportierte Prozeduren und Methoden	86
13.7. Skriptbeispiel	87
14. Modul ILTXTIN - INTERLIS 1 ohne Datenmodell lesen	88
14.1. Allgemeines	88
14.2. Abhängigkeiten von anderen Modulen	88
14.3. Parametermap	88
14.4. Objektmodell	88
14.5. Exportierte Prozeduren und Methoden	90
14.6. Skriptbeispiel	90
15. Modul LOGIN - ICS Logdateien lesen	90
15.1. Allgemeines	90
15.2. Abhängigkeiten von anderen Modulen	91
15.3. Parametermap LOGIN_PARAM	91
15.4. Objektmodell	91
15.5. Exportierte Prozeduren und Methoden	91
15.6. Skriptbeispiel	92
16. Modul MYSQLIN - MySQL lesen	93
16.1. Allgemeines	93
16.2. Abhängigkeiten von anderen Modulen	93
16.3. Parametermap DB_PARAM	93
16.4. Parametermap DBIN_PARAM	94
16.5. Parametermap MYSQLIN_PARAM	94
16.6. Objektmodell	94
16.7. Exportierte Prozeduren und Methoden	95
16.8. Skriptbeispiel	95
17. Modul ORAIN - Oracle Datenbank lesen	96
17.1. Allgemeines	96
17.2. Abhängigkeiten von anderen Modulen	96
17.3. Parametermap DB_PARAM	97
17.4. Parametermap DBIN_PARAM	97
17.5. Parametermap ORAIN_PARAM	97
17.6. Objektmodell	98
17.7. Spezielles	98
17.8. Exportierte Prozeduren und Methoden	99
17.9. Skriptbeispiel	100
18. Modul PGRESIN - PostGreSQL/PostGIS Datenbank lesen	101
18.1. Allgemeines	101
18.2. Abhängigkeiten von anderen Modulen	101
18.3. Parametermap DB_PARAM	101
18.4. Parametermap DBIN_PARAM	102
18.5. Parametermap PGRESIN_PARAM	102
18.6. Objektmodell	102
18.7. Exportierte Prozeduren und Methoden	103
18.8. Skriptbeispiel	104
19. Modul SERIALIN - GeoShop Konfigurationsdateien lesen	105
19.1. Allgemeines	105
19.2. Abhängigkeiten von anderen Modulen	105

19.3. Parametermap SERIALIN_PARAM	105
19.4. Objektmodell	105
19.5. Exportierte Prozeduren und Methoden	105
19.6. Skriptbeispiel	106
20. Modul SHPIN - ESRI Shapefile lesen	107
20.1. Allgemeines	107
20.2. Abhängigkeiten von anderen Modulen	107
20.3. Parametermap SHPIN_PARAM	107
20.4. Objektmodell	107
20.5. Exportierte Prozeduren und Methoden	108
20.6. Skriptbeispiel	108
21. Modul SQLITEIN - SQLite-Datenbank lesen	109
21.1. Allgemeines	109
21.2. Parametermap SQLITEIN_PARAM	109
21.3. Objektmodell	109
21.4. Datenbankmodell	109
21.5. Exportierte Prozeduren und Methoden	110
21.6. Skriptbeispiel	111
22. Modul TXTIN - Textdateien lesen	111
22.1. Allgemeines	111
22.2. Abhängigkeiten von anderen Modulen	111
22.3. Parametermap TXTIN_PARAM	112
22.4. Objektmodell	112
22.5. Exportierte Prozeduren und Methoden	112
22.6. Skriptbeispiel	113
23. Modul XLSIN - MS EXCEL lesen	113
23.1. Allgemeines	113
23.2. Abhängigkeiten von anderen Modulen	113
23.3. Parametermap DB_PARAM	113
23.4. Parametermap DBIN_PARAM	114
23.5. Parametermap XLSIN_PARAM	114
23.6. Objektmodell	114
23.7. EXCEL Tabelle vorbereiten	114
23.8. Datenbankmodell	116
23.9. Exportierte Prozeduren und Methoden	117
23.10. Skriptbeispiel	117
24. Modul XMLSAX - XML lesen	118
24.1. Allgemeines	118
24.2. Abhängigkeiten von anderen Modulen	118
24.3. Parametermap XMLSAX_PARAM	118
24.4. Objektmodell	119
24.5. Objektmodell bei einem SAX-Ereignis	119
24.6. Exportierte Prozeduren und Methoden	120
24.7. Skriptbeispiel	120
B. Output Module	121
1. Einleitung	121
2. Modul ARCGISOUT - ESRI Geodatabase schreiben	122
2.1. Allgemeines	122
2.2. ESRI Lizenz	122
2.3. Parametermap ARCGISOUT_PARAM	122
2.4. ArcGIS SDE Connect	123
2.5. Objektmodell	125
2.6. Record Definitionen	125
2.7. Datasets	127
2.8. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	127
2.9. Topologie Flächennetze	127
2.10. Exportierte Prozeduren und Methoden	131

2.11. Skriptbeispiel	132
2.12. Bestehende Konfigurationen IL2GDB/IL2SDE oder GDB2IL/SDE2IL nach IL2ARCGIS/ARCGIS2IL migrieren	134
3. Modul DBOUT - ODBC-Datenbank schreiben	134
3.1. Allgemeines	134
3.2. Abhängigkeiten von anderen Modulen	134
3.3. Parametermap DB_PARAM	134
3.4. Parametermap DBOUT_PARAM	135
3.5. Objektmodell	136
3.6. Record Definitionen	136
3.7. Datasets	139
3.8. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	139
3.9. Exportierte Prozeduren und Methoden	140
3.10. Skriptbeispiel	140
4. Modul DGNOUT - Bentley Microstation DGN schreiben	142
4.1. Allgemeines	142
4.2. Abhängigkeiten von anderen Modulen	142
4.3. Parametermap DGNOUT_PARAM	142
4.4. Objektmodell	142
4.5. Maps für Signaturen	145
4.6. Map für Textjustierung	146
4.7. Exportierte Prozeduren und Methoden	146
4.8. Skriptbeispiel	151
5. Modul DXFOUT - AutoCAD DXF schreiben	152
5.1. Allgemeines	152
5.2. Abhängigkeiten von anderen Modulen	152
5.3. Parametermap DXFOUT_PARAM	152
5.4. Objektmodell	153
5.5. Maps für Signaturen	155
5.6. DXF Templates	156
5.7. Exportierte Prozeduren und Methoden	157
5.8. Skriptbeispiel	159
6. Modul FILEGDBOUT - ESRI File-Geodatabase schreiben	160
6.1. Allgemeines	160
6.2. ESRI Lizenz	160
6.3. Parametermap FILEGDBOUT_PARAM	160
6.4. Objektmodell	160
6.5. Record Definitionen	161
6.6. Datasets	163
6.7. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	163
6.8. Exportierte Prozeduren und Methoden	163
6.9. Skriptbeispiel	164
7. Modul GEOJSONOUT - GeoJSON schreiben	165
7.1. Allgemeines	165
7.2. Abhängigkeiten von anderen Modulen	165
7.3. Parametermap GEOJSONOUT_PARAM	165
7.4. Objektmodell	166
7.5. Exportierte Prozeduren und Methoden	166
7.6. Skriptbeispiel	166
8. Modul GMMDBOUT - Intergraph GeoMedia ACCESS Datenbank schreiben	168
8.1. Allgemeines	168
8.2. Abhängigkeiten von anderen Modulen	169
8.3. Parametermap DB_PARAM	169
8.4. Parametermap DBOUT_PARAM	169
8.5. Parametermap GMMDBOUT_PARAM	169
8.6. Objektmodell	170

8.7. Record Definitionen	170
8.8. Datasets	172
8.9. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	172
8.10. Exportierte Prozeduren und Methoden	172
8.11. Skriptbeispiel	174
9. Modul GMORAOUT - Intergraph GeoMedia Oracle Datenbank schreiben	176
9.1. Allgemeines	176
9.2. Abhängigkeiten von anderen Modulen	176
9.3. Parametermap DB_PARAM	176
9.4. Parametermap DBOUT_PARAM	177
9.5. Parametermap ORAOUT_PARAM	177
9.6. Parametermap GMORAOUT_PARAM	179
9.7. Objektmodell	180
9.8. Record Definitionen	180
9.9. Datasets	182
9.10. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	182
9.11. Exportierte Prozeduren und Methoden	182
9.12. Skriptbeispiel	184
10. Modul GMSQLOUT - Intergraph GeoMedia SQL Server Datenbank schrei- ben	187
10.1. Allgemeines	187
10.2. Abhängigkeiten von anderen Modulen	187
10.3. Parametermap DB_PARAM	187
10.4. Parametermap DBOUT_PARAM	188
10.5. Parametermap GMSQLOUT_PARAM	188
10.6. Objektmodell	188
10.7. Record Definitionen	189
10.8. Datasets	190
10.9. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	190
10.10. Exportierte Prozeduren und Methoden	190
10.11. Skriptbeispiel	192
11. Modul IFCOUT - Industry Foundation Classes IFC schreiben	194
11.1. Allgemeines	194
11.2. Parametermap IFCOUT_PARAM	194
11.3. Objektmodell	194
11.4. Exportierte Prozeduren und Methoden	196
11.5. Skriptbeispiel	197
12. Modul IL2OUT - INTERLIS 2 schreiben	198
12.1. Allgemeines	198
12.2. Abhängigkeiten von anderen Modulen	199
12.3. Parametermap IL2OUT_PARAM	199
12.4. Objektmodell	200
12.5. Exportierte Prozeduren und Methoden	200
12.6. Skriptbeispiel	201
13. Modul ILOUT - INTERLIS 1 schreiben	202
13.1. Allgemeines	202
13.2. Abhängigkeiten von anderen Modulen	202
13.3. Parametermap ILOUT_PARAM	202
13.4. Objektmodell	202
13.5. Exportierte Prozeduren und Methoden	203
13.6. Skriptbeispiel	204
14. Modul KMLOUT - Google KML schreiben	205
14.1. Allgemeines	205
14.2. Abhängigkeiten von anderen Modulen	205
14.3. Parametermap KMLOUT_PARAM	205

14.4. Objektmodell	206
14.5. Record Definitionen	206
14.6. Folder Definitionen	207
14.7. KML Templates	208
14.8. Exportierte Prozeduren und Methoden	208
14.9. Skriptbeispiel	211
15. Modul MYSQLOUT - MySQL-Datenbank schreiben	213
15.1. Allgemeines	213
15.2. Abhängigkeiten von anderen Modulen	213
15.3. Parametermap DB_PARAM	213
15.4. Parametermap DBOUT_PARAM	214
15.5. Parametermap MYSQLOUT_PARAM	214
15.6. Objektmodell	215
15.7. Record Definitionen	215
15.8. Datasets	216
15.9. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	217
15.10. Prozeduren und Methoden	217
15.11. Skriptbeispiel	218
16. Modul ORAOUT - Oracle-Datenbank schreiben	220
16.1. Allgemeines	220
16.2. Abhängigkeiten von anderen Modulen	220
16.3. Parametermap DB_PARAM	220
16.4. Parametermap DBOUT_PARAM	221
16.5. Parametermap ORAOUT_PARAM	222
16.6. Objektmodell	223
16.7. Record Definitionen	223
16.8. Datasets	225
16.9. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	225
16.10. Prozeduren und Methoden	225
16.11. Skriptbeispiel	227
17. Modul PGRESOUT - PostGreSQL/PostGIS-Datenbank schreiben	228
17.1. Allgemeines	228
17.2. Abhängigkeiten von anderen Modulen	229
17.3. Parametermap DB_PARAM	229
17.4. Parametermap DBOUT_PARAM	229
17.5. Parametermap PGRESOUT_PARAM	230
17.6. Objektmodell	230
17.7. Record Definitionen	231
17.8. Datasets	232
17.9. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	232
17.10. Prozeduren und Methoden	233
17.11. Skriptbeispiel	234
18. Modul PSOUT - PostScript (PDF,JPEG,TIF) schreiben	236
18.1. Allgemeines	236
18.2. Abhängigkeiten von anderen Modulen	236
18.3. Parametermap PSOUT_PARAM	236
18.4. Symbole Clippen	237
18.5. Signaturen	238
18.6. True Type Fonts	238
18.7. GhostScript	239
18.8. Objektmodell	240
18.9. Exportierte Prozeduren und Methoden	241
18.10. Skriptbeispiel	242
19. Modul SHPOUT - ESRI Shapefile schreiben	244
19.1. Allgemeines	244

19.2. Abhängigkeiten von anderen Modulen	244
19.3. Parametermap SHPOUT_PARAM	245
19.4. Objektmodell	245
19.5. Map für Textsignaturen	246
19.6. Exportierte Prozeduren und Methoden	247
19.7. Skriptbeispiel	248
20. Modul SQLITEOUT - SQLite-Datenbank schreiben	249
20.1. Allgemeines	249
20.2. Parametermap SQLITEOUT_PARAM	250
20.3. Objektmodell	250
20.4. Record Definitionen	250
20.5. Datasets	252
20.6. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	252
20.7. Exportierte Prozeduren und Methoden	253
20.8. Skriptbeispiel	253
21. Modul TXTOUT - Textdateien schreiben	255
21.1. Allgemeines	255
21.2. Abhängigkeiten von anderen Modulen	255
21.3. Parametermap TXTOUT_PARAM	255
21.4. Objektmodell	255
21.5. Record Definitionen	255
21.6. Exportierte Prozeduren und Methoden	256
21.7. Skriptbeispiel	257
22. Modul XLSOUT - MS Excel schreiben	258
22.1. Allgemeines	258
22.2. Abhängigkeiten von anderen Modulen	259
22.3. Parametermap DB_PARAM	259
22.4. Parametermap DBOUT_PARAM	259
22.5. Parametermap XLSOUT_PARAM	260
22.6. Objektmodell	260
22.7. EXCEL Tabelle vorbereiten	260
22.8. Record Definitionen	262
22.9. Datasets	263
22.10. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MO- DEL	263
22.11. Exportierte Prozeduren und Methoden	264
22.12. Skriptbeispiel	264
C. Verarbeitungs Module	266
1. Einleitung	266
2. Modul GEOINDEX - Geometrie Index	266
2.1. Allgemeines	266
2.2. Parametermap GEOINDEX_PARAM	266
2.3. Objektmodell	267
2.4. Exportierte Prozeduren und Methoden	267
2.5. Skriptbeispiel	270
3. Modul INTERSECT - Verschnitt Flächen, Linien, Punkte	271
3.1. Allgemeines	271
3.2. Abhängigkeiten von anderen Modulen	271
3.3. Beispiel	272
3.3.1. Ausgangslage	272
3.3.2. Verschnitt Modus MAININTERSECTION	273
3.3.3. Verschnitt Modus INTERSECTION	274
3.4. Funktionsweise	274
3.5. Hilfskonfiguration	275
3.6. Parametermap INTERSECT_PARAM	275
3.7. Objektmodell	275
3.8. Exportierte Prozeduren und Methoden	277

3.9. Skriptbeispiel	278
4. Modul LIST - ICS Objekte temporär speichern	280
4.1. Allgemeines	280
4.2. Abhängigkeiten von anderen Modulen	280
4.3. Parametermap	280
4.4. Objektmodell	280
4.5. Exportierte Prozeduren und Methoden	280
4.6. Skriptbeispiel	281
5. Modul MAP - ICS Objekte temporär speichern	281
5.1. Allgemeines	281
5.2. Abhängigkeiten von anderen Modulen	282
5.3. Parametermap	282
5.4. Objektmodell	282
5.5. Exportierte Prozeduren und Methoden	282
5.6. Skriptbeispiel	282
6. Modul MTOPO - Mehrere AREA Topologien berechnen	283
6.1. Allgemeines	283
6.2. Abhängigkeiten von anderen Modulen	283
6.3. Parametermap TOPO_PARAM	284
6.4. Objektmodell	284
6.5. Exportierte Prozeduren und Methoden	284
6.6. Skriptbeispiel	285
7. Modul NOOP - Spezielle Initialisierungen	286
7.1. Allgemeines	286
7.2. Abhängigkeiten von anderen Modulen	286
7.3. Parametermap	286
7.4. Objektmodell	286
7.5. Exportierte Prozeduren und Methoden	286
7.6. Skriptbeispiel	286
8. Modul OSTREAM - ICS Objekte temporär speichern	287
8.1. Allgemeines	287
8.2. Abhängigkeiten von anderen Modulen	287
8.3. Parametermap	287
8.4. Objektmodell	288
8.5. Exportierte Prozeduren und Methoden	288
8.6. Skriptbeispiel	289
9. Modul PLOT - Plotlayout schreiben	289
9.1. Allgemeines	289
9.2. Abhängigkeiten von anderen Modulen	290
9.3. Parametermap PLOT_PARAM	290
9.4. Plotlayout Map PLOT_LAYOUT	292
9.5. Koordinatenkreuze Map PLOT_COORDCROSS_WIDTH	292
9.6. Skalierungsband Map PLOT_SCALEBAND_WIDTH	293
9.7. Werte Map PLOT_VALUES	294
9.8. Objekt Map PLOT_WRITE_OBJECT	294
9.9. Anwendung	295
9.10. Exportierte Prozeduren und Methoden	295
9.11. Skriptbeispiel	295
10. Modul STAT - Statistiken aus INTERLIS Daten erzeugen	301
10.1. Allgemeines	301
10.2. Abhängigkeiten von anderen Modulen	301
10.3. Parametermap STAT_PARAM	301
10.4. Objektmodell	301
10.5. Exportierte Prozeduren und Methoden	301
10.6. Skriptbeispiel	302
10.7. Beispiel für Statistikdatei	303
11. Modul SURFCUT - Flächenverschnitt	303
11.1. Allgemeines	303

11.2. Anpassung eines Script vom Modul SURFCUT auf das Modul INTERSECT.	303
12. Modul TOPO - Topologie berechnen	304
12.1. Allgemeines	304
12.2. Abhängigkeiten von anderen Modulen	304
12.3. Parametermap TOPO_PARAM	304
12.4. Objektmodell	304
12.5. Exportierte Prozeduren und Methoden	305
12.6. Skriptbeispiel	306
13. Modul VPRI0 - Vektor Elimination nach Prioritäten	307
13.1. Allgemeines	307
13.2. Abhängigkeiten von anderen Modulen	307
13.3. Parametermap VPRI0_PARAM	307
13.4. Objektmodell	308
13.5. Exportierte Prozeduren und Methoden	308
13.6. Skriptbeispiel	309
D. iG/Script Bibliotheken	310
1. Einleitung	310
2. Skriptbibliothek OS.LIB	310
2.1. Allgemeines	310
2.2. Exportierte Prozeduren	311
3. Skriptbibliothek UTIL.LIB	311
3.1. Allgemeines	311
3.2. Exportierte Prozeduren	311
4. Skriptbibliothek TRANSFORM.LIB	314
4.1. Allgemeines	314
4.2. Parametermap TRANSFORM_PARAM	314
4.3. Exportierte Prozeduren	315
4.4. Skriptbeispiel	315

1. Allgemeiner Aufbau von ICS Konfigurationen

1.1. infoGrips Conversion System (ICS)

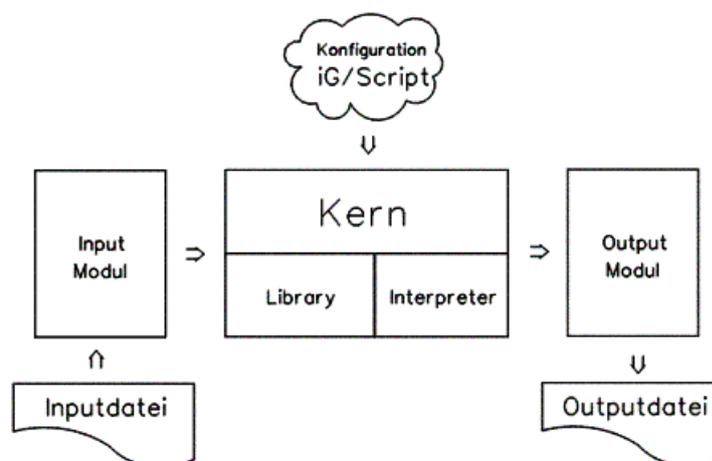
Schnittstellen für den Austausch von GIS-Daten (GIS = Geographisches Informations System) werden heute meist als C/C++, BASIC- oder FORTRAN-Programme kodiert. Der Programmieraufwand für die Erstellung einer Schnittstelle ist vor allem im Bereich der GIS-Systeme gross, da neben Sachdaten auch Geometriedaten übertragen werden müssen. Dies führt dazu, dass von den GIS-Systemherstellern nur relativ wenige Schnittstellen (häufig DXF) mit begrenztem Funktionsumfang angeboten werden.

Mit der Entwicklung von ICS wurde ein anderer Weg beschritten. Bei ICS handelt es sich um ein erweiterbares Schnittstellensystem für die schnelle Entwicklung von GIS-Schnittstellen. Folgende Ideen lagen der Entwicklung von ICS zugrunde:

- Jedes **Schnittstellenprogramm** kann in einen **Input-** und in einen **Outputmodul** zerlegt werden. Dabei liest der Inputmodul Objekte aus der Inputdatei und wandelt sie in ein neutrales internes Objektformat um. Der Outputmodul nimmt Objekte vom Inputmodul entgegen und schreibt sie in die Outputdatei. Damit wird eine Entkoppelung des Schnittstellenprogramms in zwei unabhängige, wiederverwendbare Module erreicht.
- Im **Kern** werden die Funktionen zusammengefasst, die von allen Modulen benötigt werden (z.B. String- und Geometriefunktionen). Dadurch müssen diese Funktionen nur einmal programmiert werden.
- Der Datenfluss der Objekte vom Inputmodul zum Outputmodul wird nicht durch ein fix kodiertes Programm gesteuert, sondern über die Skriptsprache **iG/Script**.

Nachfolgend ist die Architektur einer ICS-Schnittstelle dargestellt:

Abbildung 1. ICS Architektur



Der Kern und die Input- bzw. Outputmodule werden von der infoGrips GmbH entwickelt. iG/Script-Konfigurationen können auch vom Benutzer geschrieben werden.

Der ICS-Kern enthält neben den allgemeinen Funktionen für die Behandlung von Datenstrukturen (Strings, Geometrie, Maps etc.) einen Interpreter für die Sprache iG/Script. iG/Script ist eine allgemeine Programmiersprache mit einem vordefinierten Satz von Standardfunktionen. Die Sprache enthält neben arithmetischen-, logischen- und Zuweisungsoperationen, Kontrollstrukturen wie IF und WHILE. Daneben bietet sie die Möglichkeit, den Sprachumfang durch Prozeduren zu erweitern. Als Basistypen kennt die Sprache iG/Script die Typen Integer, Real, String, Boolean und Geometrie. Strukturierte Datentypen können über den Datentyp Map und List erzeugt werden. Input- bzw. Outputmodule können, falls nötig, zusätzliche Datentypen implementieren.

Zu bestehenden Programmiersprachen ist iG/Script am ehesten mit der Programmiersprache FORTH verwandt. Mit FORTH verbindet sie, dass sie ebenfalls alle Operationen über einen **Stack** abwickelt und eine klammerfreie Darstellung von Ausdrücken verwendet. Der wesentliche Unterschied zwischen FORTH und iG/Script liegt darin, dass FORTH für die hardwarenahe Programmierung entwickelt wurde, iG/Script hingegen ist eine hardwareunabhängige Sprache, die sich besonders für die Entwicklung von Schnittstellenapplikationen eignet.

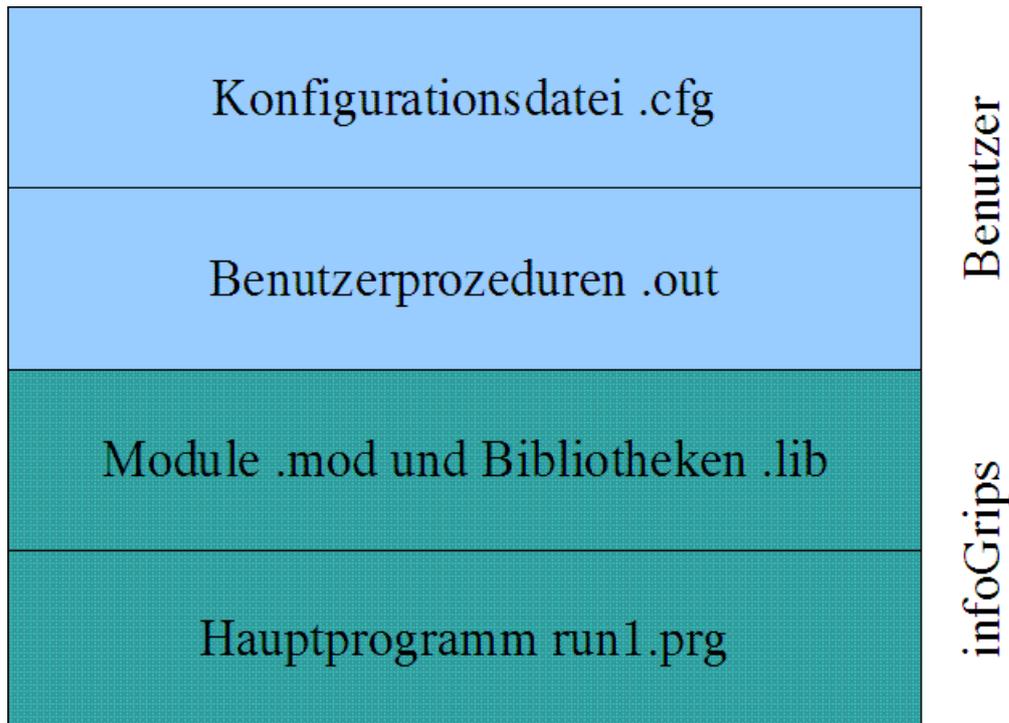
iG/Script ist im **iG/Script Benutzer- und Referenzhandbuch** detailliert beschrieben.

1.2. Grundstruktur von ICS Konfigurationen

Jeder ICS Konfigurationsscript besteht aus folgenden Skriptteilen:

- Der .cfg Konfigurationsdatei, welche alle Konfigurationsparameter enthält und den Datenfluss der Schnittstelle regelt.
- Den (optionalen) .out Dateien, welche benutzerdefinierte iG/Script Prozeduren enthalten.
- Den .lib Dateien, welche nützliche Sammlungen von iG/Script Prozeduren enthalten.
- Den .mod Dateien, welche Input-, Output- oder Verarbeitungsmodule implementieren.
- Dem Verarbeitungsalgorithmus RUN1.

Will ein Benutzer eine ICS Konfiguration erstellen, so muss er eine .cfg Datei und allenfalls eine oder mehrere .out Datei erzeugen. Die übrigen Konfigurationsteile (d.h. .lib, .mod und .prg Dateien) werden von der infoGrips GmbH zur Verfügung gestellt, und sind vollständig in diesem Handbuch dokumentiert. Nachfolgend ist die Aufteilung einer ICS Konfiguration in ihre Skriptteile dargestellt:

Abbildung 2. Grundstruktur von ICS Konfigurationen

Die benutzerdefinierbaren Konfigurationsteile sind in den nächsten Abschnitten beschrieben. Die Beschreibung der von der infoGrips GmbH zur Verfügung gestellten Komponenten ist in den Anhängen dieser Dokumentation enthalten.

1.3. Inhalt der .cfg Datei

Die .cfg Datei enthält folgende Konfigurationsteile:

- Die Angabe der Lizenzinformationen.
- Die Definition des Benutzerinputs.
- Die Parameter für die Input-, Output- bzw. Verarbeitungsmodule.
- Die Definition der Inputquellen.
- Die Festlegung des Verarbeitungsablaufs.
- Die Angabe der benötigten Skriptbibliotheken.
- Die Angabe der benötigten Input-, Output- und Verarbeitungsmodule.
- Die Angabe der verwendeten Benutzerprozeduren.
- Die Angabe des Verarbeitungsalgorithmus RUN1.

Nachfolgend ist ein typisches Beispiel einer .cfg Datei dargestellt, an welchem die einzelnen Konfigurationsteile kurz vorgestellt werden:

```
!=====!  
!  
! INTERLIS => TEXT Configuration Vers. 1.0  
!
```

```
!=====!  
  
!<1>  
|LICENSE \license\iltoolspro.lic  
|LICENSE \license\iltools.lic  
  
!+++++!  
!  
! user input  
!  
  
!<2a>  
MAP USER_INPUT1  
  DIALOG => FILE ! FILE | STRING | ODBC  
  MESSAGE => 'Enter .itf Input File'  
  FILE_FILTER => itf  
  FILE_EXISTS => TRUE  
  OPT => input  
END_MAP  
  
!<2b>  
MAP USER_INPUT2  
  DIALOG => FILE ! FILE | STRING | ODBC  
  MESSAGE => 'Enter .txt Output File'  
  FILE_FILTER => txt  
  FILE_EXISTS => FALSE  
  OPT => output  
END_MAP  
  
!+++++!  
!  
! parameter maps for input modules  
!  
  
!<3a>  
MAP ILIN_PARAM  
  INTERLIS_DEF => \models\Grunddatensatz.ili  
  LOG_TABLE => ON  
  TRACE => OFF  
  STATISTICS => ON  
  VALUE_CHECK => OFF  
  ENUM_TO_TEXT => ON  
END_MAP  
  
!+++++!  
!  
! parameter maps for output modules  
!  
  
!<3b>  
MAP TXTOUT_PARAM  
  STATISTICS => ON  
END_MAP  
  
!+++++!  
!  
! input sources  
!
```

```

!<4>
MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
END_MAP

!+++++
!
! classification
!

!<5>
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => Fix_Punkt0
  I1,Fixpunkte,LFP3 => Fix_Punkt0
  I1,* => OFF
END_MAP

!<6>
|INCL \script\util.lib
!<7>
|INCL \script\ilin.mod
|INCL \script\txtout.mod
!<8>
|INCL \script\il2txt\Grunddatensatz.out
!<9>
|INCL \script\run1.prg

```

Erläuterungen zu den einzelnen Konfigurationsteilen:

1. Mit der |LICENSE Direktive werden die von der Konfiguration benötigten Lizenzen angegeben. Es können mehrere Lizenzdateien angegeben werden, wenn die Konfiguration unter verschiedenen Lizenzen lauffähig ist (z.B. INTERLIS Tools und INTERLIS Tools Professional). Falls diese Angaben vergessen werden, bricht die Konfiguration zur Laufzeit mit einer Fehlermeldung ab (no license found for ...).
2. Die meisten Konfigurationen verlangen vom Benutzer interaktive Eingaben (z.B. Auswahl der Inputdatei bzw. Angabe der Outputdatei). Mit den USER_INPUTx Maps können beliebig viele Parameter interaktiv vom Benutzer abgefragt werden (s.a. ???).
3. Alle Input-, Output- und Verarbeitungsmodul benötigen diverse Parameter, z.B. die Angabe des Datenmodells für einen INTERLIS 1 Outputmodul. Die Parameter für die diversen von der Konfiguration benötigten Module werden in den <MODUL>_PARAM Maps definiert. Alle Parameter der einzelnen Module sind im Anhang beschrieben. Einige Module benötigen neben der <MODUL>_PARAM Map noch weitere Angaben (z.B. Symbologie Maps für den DXFOUT Modul). In diesem Fall ist das ebenfalls in der Moduldokumentation im Anhang beschrieben.
4. In der Map INPUT_SOURCES wird festgelegt, in welcher Reihenfolge die Inputmodule ausgelesen werden (s.a. ???).
5. Die INOUT Map legt fest, wie die von den Inputmodulen gelesenen Objekte an die Verarbeitungs- bzw. Outputmodule weiter geleitet werden (s.a. ???).
6. Skriptbibliotheken werden mit der |INCL Direktive eingebunden. Die verfügbaren Skriptbibliotheken sind im Anhang beschrieben.

7. Alle benötigten Input-, Output- und Verarbeitungsmodulare müssen mit `| INCL` eingebunden werden. Die verfügbaren Module sind im Anhang beschrieben.
8. Falls die Konfiguration benutzerdefinierte Prozeduren verwendet, müssen diese mit `| INCL` eingebunden werden (s.a. ???).
9. Schliesslich muss mit `| INCL \script\run1.prg` der Verarbeitungsalgorithmus angegeben werden, welcher den Inhalt der `.cfg` Datei interpretiert. Im Moment steht der `RUN1` Algorithmus zur Verfügung (s.a. ???).

1.4. Der RUN1 Algorithmus

Der `RUN1` Algorithmus interpretiert den Inhalt der `.cfg` Datei und steuert die Input-, Output- und Verarbeitungsmodulare. Der `RUN1` Algorithmus ist in der Skriptdatei `\script\run1.prg` implementiert. Diese Datei muss daher am Ende jeder `.cfg` Datei mit `| INCL` eingebunden werden. Der `RUN1` Algorithmus verfügt über folgende Eigenschaften:

- Kann beliebig viele Parameter vom Benutzer interaktiv abfragen.
- Kann beliebig viele Inputquellen verarbeiten.
- Kann beliebig viele Input-, Output- und Verarbeitungsmodulare über die `INOUT` Map verknüpfen.
- Kann Macros aus der Map `MACRO` verarbeiten.
- Kann parametrisierte Benutzerprozeduren aufrufen.
- Kann Triggerprozeduren aufrufen.

Die Einzelnen Eigenschaften von `RUN1` werden in den folgenden Unterabschnitten näher erläutert.

1.4.1. Abfrage von Benutzerparametern

Sollen in einer Konfiguration interaktiv Parameter abgefragt werden (z.B. Input- und Outputdatei), so müssen in der `.cfg` Datei Maps der Form `USER_INPUTx` angelegt werden, wobei `x` für eine Zahl zwischen 1 .. 9 steht. Die `USER_INPUTx` Maps werden vom `RUN1` Algorithmus in der Reihenfolge ihrer Nummer ausgewertet (zuerst 1, dann 2, dann 3, etc.). Nachfolgend ist ein Beispiel für eine `USER_INPUT` Map dargestellt:

```
MAP USER_INPUT2
  DIALOG => FILE ! FILE | FILES | ZIP | STRING | ODBC
  MESSAGE => 'Enter .txt Output File'
  FILE_FILTER => txt
  FILE_EXISTS => FALSE
  OPT => output
END_MAP
```

Das obige Beispiel fragt vom Benutzer eine Datei ab (`DIALOG => FILE`), welche nicht bereits existiert (`FILE_EXISTS => FALSE`) und die Endung `.txt` aufweisen muss (`FILE_FILTER => txt`). Die einzelnen Parameter der `USER_INPUT` Maps haben folgende Bedeutung:

DIALOG

FILE

Der Benutzer muss eine Datei auswählen. Ob die Datei bereits existieren muss oder nicht, wird mit dem Parameter `FILE_EXISTS` (`TRUE` | `FALSE`) festgelegt.

Weiter kann mit `FILE_FILTER` angegeben werden, welche Dateieendung die Datei aufweisen muss.

FILES

Im Prinzip gleich wie `FILE` mit dem Unterschied, dass mehrere Dateien vom Benutzer ausgewählt werden können.

ZIP

Erweiterung zu `FILE`. Es muss ein `.zip` Archiv ausgewählt werden. In `ZIP_FILTER` kann zusätzlich ein Filter für die aus dem Archiv zu selektierenden Dateien angegeben werden.

DIRECTORY

Der Benutzer muss ein Dateiverzeichnis auswählen.

STRING

Der Benutzer muss eine Zeichenkette (String) eingeben.

ODBC

Der Benutzer muss eine bestehende ODBC Datenquelle auswählen.

ODBC_FILE

Der Benutzer muss eine bestehende ODBC Datenquelle oder ein Datenbankfile auswählen.

ARCGIS

Der Benutzer muss eine bestehende ArcGIS SDE Connection oder ein ArcGIS Datenbankfile auswählen.

PROCEDURE

Es wird eine vom Benutzer unter `PROCEDURE_NAME` angegebene Prozedur aufgerufen, welche den Parameterwert abfragt. Die Prozedur muss als Resultat einen String oder eine Liste von Strings auf dem Stack zurück liefern.

CONSTANT

Der Parameterwert ist eine Konstante. Der Parameterwert kann in `OPT_VALUE` gesetzt werden.

MESSAGE

Meldung welche dem Benutzer bei der Abfrage des Parameters angezeigt werden soll. Bemerkung: Falls die Meldung Leerzeichen enthält, muss die Meldung zwischen Hochkommas gestellt werden.

OPT

Name unter welchem der Wert des Parameters in der OPT Map abgespeichert werden soll.

1.4.2. Inputquellen

Von einer ICS Konfiguration können gleichzeitig mehrere Inputquellen verarbeitet werden. Inputquellen lesen Objekte von einer externen Datenquelle (z.B. INTERLIS-Datei, ODBC-Datenbank) und wandeln diese Objekte in eine interne Struktur um (IN-Objekt). Der Benutzer muss die von ihm benötigten Inputquellen in der Map `INPUT_SOURCES` definieren.

Beispiel 1. Definition von Inputquellen

```
MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
  D1 => DB,features
END_MAP
```

Erläuterungen zum obigen Beispiel:

- In der Map `INPUT_SOURCES` wurden die Inputquellen `I1` und `D1` definiert.
- Die Quelle `I1` liest Inputobjekte aus der vom Benutzer gewählten INTERLIS-Inputdatei (`OPT.input`) welche in `ILIN_PARAM` konfiguriert wurde.
- Die Quelle `D1` liest Datensätze aus der Tabelle `features` einer relationalen Datenbank, die in `DB_PARAM` konfiguriert wurde.

Es ist erlaubt beliebig viele Inputquellen (z.B. mehrere INTERLIS-Quellen und mehrere Datenbankquellen) gleichzeitig zu definieren. Die Inputquellen werden von `RUN1` in der Reihenfolge geöffnet und gelesen, in der sie definiert wurden (hier: zuerst `I1`, dann `D1`).

1.4.3. Die INOUT Map

Alle von den Inputquellen gelieferten Objekte müssen klassifiziert werden. Die Klassifikation wird nach folgendem Schema durch Einträge in der Map `INOUT` ermittelt:

1. Die erste Klassifikation ist der Name der gerade aktiven Inputquelle (z.B. `I1`).
2. In den folgenden Schritten wird versucht, die Klassifikation zu verfeinern. Dazu wird die aktuelle Klassifikation erneut durch die Map `INOUT` abgebildet.
3. Falls die Abbildung eine Abbildungsvorschrift liefert, ist die Klassifikation abgeschlossen. Das Resultat der Klassifikation ist die Abbildungsvorschrift (z.B. `COPY`). Weiter mit Schritt 7.
4. Falls die Abbildung den Namen einer IN-Objekt Komponente liefert (z.B. `IN.TABLE`) wird der Wert (z.B. `BoFlaeche`) der Komponente an die aktuelle Klassifikation durch eine Komma getrennt angehängt. Weiter mit Schritt 2.
5. Falls die Klassifikation in der Map `INOUT` keine Abbildung hat, wird der letzte Teil der aktuellen Klassifikation durch `*` ersetzt. Weiter mit Schritt 2.
6. Falls noch immer keine Abbildung gefunden werden konnte, ist dies ein Fehler. Das Objekt wird in die Map `UNDEFINED_OBJECTS` eingetragen. Fertig.
7. Falls die Abbildungsvorschrift gleich `OFF` ist, wird das Objekt ignoriert (bzw. in die Map `IGNORED_OBJECTS` eingetragen), sonst wird die gefundene Abbildungsvorschrift ausgeführt.

Im folgenden Beispiel werden Objekte aus der INTERLIS Tabelle `Fixpunkte.HFP` je nach Inhalt der Komponente `IN.TOPIC` und `IN.TABLE` klassifiziert.

Beispiel 2. INOUT Map

```
MAP INOUT
  I1 => IN.TOPIC, IN.TABLE
  I1,Fixpunkte,HFP => COPY_IN_OUT0
END_MAP
```

Erläuterungen:

- Da die Daten von der Inputquelle `I1` gelesen werden, ist der Wert der ersten Klassifikation für jedes Objekt `I1` (Schritt 1).
- `I1` wird durch `INOUT` abgebildet das Resultat ist `IN.TOPIC, IN.TABLE` (Schritt 2).

- Da es sich bei `IN.TOPIC`, `IN.TABLE` um Komponenten des IN-Objekts handelt, wird der Inhalt von `IN.TOPIC`, `IN.TABLE` durch ein Komma getrennt an die aktuelle Klassifikation angehängt (hier: `I1,Fixpunkte,HFP`)(Schritt 4).
- Die aktuelle Klassifikation wird wieder durch `INOUT` abgebildet (Schritt 2) und man erhält schliesslich `COPY_IN_OUT0` (Schritt 3).
- `COPY_IN_OUT0` ist eine Abbildungsvorschrift, die von ICS ausgeführt wird.

Man kann das obige Beispiel auch als eine speziell kodierte IF Anweisung verstehen. Der iG/Script Code dazu lautet wie folgt:

```
IF VAR.SOURCE = 'I1' THEN
  IF IN.TOPIC = 'Fixpunkte' THEN
    IF IN.TABLE = 'HFP' THEN
      COPY_IN_OUT0
    END_IF
  END_IF
END_IF
```

1.4.4. Ausführen der Abbildungsvorschrift

Die in ??? gefundenen Abbildungsvorschriften werden nach folgendem Schema ausgeführt:

1. Die Abbildungsvorschrift ist eine Liste von iG/Script Prozeduren und/oder Macros.
2. Alle Macros werden durch ihren Wert ersetzt (Macros können in der Map `MACRO` deklariert werden, s.a. 4.1.7).
3. Falls das erste Element der Liste eine iG/Script Prozedur ist, wird die Prozedur aufgerufen. Weiter mit Schritt 5.
4. Falls das erste Element der Liste keine iG/Script Prozedur ist, ist dies ein Fehler (unknown procedure ...). Fertig.
5. Die Prozedur wird ausgeführt und die aktuelle Abbildungsvorschrift wird um die Prozedur und ihre Argumente reduziert.
6. Falls die Abbildungsvorschrift nach der Reduktion leer ist, sind wir fertig.
7. Falls die Liste nach der Reduktion nicht leer ist, weiter mit Schritt 3.

1.4.5. Die MACRO Map

In der Map `MACRO` können Sie Abkürzungen für häufig gebrauchte Prozeduraufrufe und/oder Argumente definieren.

Beispiel 3. MACRO Map

```
MAP MACRO
  DIN => DISPLAY_OBJECT1, IN
END_MACRO
```

Hier wurde z.B. der Macro `DIN` definiert. Der Macro wird zur Laufzeit durch seinen Wert ersetzt (hier: `DISPLAY_OBJECT, IN`). Der Macro `DIN` kann nun in jeder Abbildungsvorschrift der `INOUT` Map benutzt werden.

1.4.6. Benutzerprozeduren in Abbildungsvorschriften

Der Benutzer kann ICS Konfigurationen mit eigenen iG/Script Prozeduren ergänzen. Die neuen Prozeduren sollten in einer .out Datei definiert werden. Die benutzerdefinierten Prozeduren können in jeder Abbildungsvorschrift benutzen werden. Es ist sogar möglich Prozeduren mit Parametern zu definieren. Die Programmiersprache iG/Script ist ausführlich im **iG/Script Benutzer- und Referenzhandbuch** beschrieben.

Benutzerprozeduren müssen ihre Parameter mit den vordefinierten Prozeduren:

GET_SPARAM

String Parameter.

GET_IPARAM

Integer Parameter.

GET_RPARAM

Real Parameter.

GET_PPARAM

Punkt Parameter.

GET_LPARAM

Linien Parameter.

GET_APARAM

Flächen Parameter.

GET_PARAM

Beliebiges ICS-Objekt.

Diese Prozeduren sind in der Skriptbibliothek `\script\util.lib` definiert.

Beispiel 4. Parameterübernahme in Benutzerprozeduren

```
PROCEDURE Bodb_Centroid1 ! Art
  'Bodenbedeckung' => OUT.TOPIC
  'BoFlaeche' => OUT.TABLE
  IN.OBJID => OUT.OBJID
  IN.Geometrie => OUT.Geometrie
  GET_SPARAM => OUT.Art
  WRITE_OBJECT
END_PROCEDURE
```

Die Prozedur `Bodb_Centroid1` schreibt eine Bodenbedeckungszentroid in die INTERLIS Transferdatei. Die Prozedur übernimmt den Artcode (Art) des Zentroids als Parameter. Die neue Prozedur kann wie folgt in einer Abbildungsvorschrift der INOUT Map benutzt werden:

```
MAP INOUT
  ...
  I1,Bodenbedeckung,BoFlaeche => Bodb_Centroid1,befestigt.Bahn
  ...
END_MAP
```

Beim Ausführen der Abbildungsvorschrift wird die Prozedur `Bodb_Centroid1` mit dem Parameter `befestigt.Bahn` aufgerufen und damit ein Centroid in der INTERLIS-Tabelle `BoFlaeche` erzeugt.



Die Zahl 1 am Schluss des Namens der Benutzerprozedur, gibt an, dass die Benutzerprozedur einen Parameter verlangt. Die Angabe der Anzahl Parameter im Benutzerprozedurnamen ist *obligatorisch*. Falls die Prozedur keine Parameter übernimmt, muss 0 angegeben werden.

1.4.7. Triggerprozeduren

Triggerprozeduren sind Prozeduren, welche in einem bestimmten Moment in der Konfiguration automatisch aufgerufen werden. Der RUN1 Algorithmus unterstützt folgende Triggerprozeduren:

PRE_TRANSFER

Wird einmal vor dem Öffnen aller Quellen aufgerufen.

PRE_SOURCE_<SOURCE>

Wird vor dem Öffnen der Source <SOURCE> aufgerufen.

PRE_OPEN_<SOURCE>

Wird aufgerufen bevor Objekte aus der Source <SOURCE> ausgelesen werden. Die <SOURCE> ist aber bereits offen (im Gegensatz zu PRE_SOURCE_<SOURCE>).

 In älteren Versionen der INTERLIS Tools wurde PRE_OPEN_<SOURCE> vor PRE_SOURCE_<SOURCE> aufgerufen. Falls alte Konfigurationen von dieser Reihenfolge der Trigger abhängig sind, müssen sie auf die neue Reihenfolge umgestellt werden.

 Falls mit der in der USER_INPUTx Map der Dialog FILES ausgewählt wurde, wird PRE_OPEN_<SOURCE> und POST_CLOSE_<SOURCE> für jede ausgewählte Datei einmal aufgerufen.

PRE_INOUT_<SOURCE>

Wird vor dem Durchlaufen der INOUT Map aufgerufen. Dieser Trigger kann z.B. für die Berechnung von virtuellen Attributen benutzt werden.

 In älteren Versionen der INTERLIS Tools hiess dieser Trigger POST_READ_<SOURCE>. Aus Kompatibilitätsgründen ist daher POST_READ_<SOURCE> weiterhin mit der gleichen Bedeutung verfügbar.

POST_INOUT_<SOURCE>

Wird nach dem Durchlaufen der INOUT Map aufgerufen.

POST_CLOSE_<SOURCE>

Wird vor dem Schliessen der Source <SOURCE> aufgerufen.

POST_SOURCE_<SOURCE>

Wird nach dem Schliessen der Source <SOURCE> aufgerufen.

POST_TRANSFER

Wird einmal nach dem Schliessen aller Quellen aufgerufen.

Den gleichen Effekt wie mit Triggern kann man z.T. auch durch Verwendung der Pseudoinputquelle NOOP erreichen. NOOP ist eine spezielle Datenquelle die nur ein einziges Objekt liefert. Diese Eigenschaft kann man z.B. wie folgt nutzen:

```
MAP INPUT_SOURCES
  N1 => NOOP
  M1 => MSIN,OPT.input
END_MAP
MAP INOUT
  N1 => MeineInitialisierung0
  M1 => ... ! etc.
END_MAP
```

Die Inputquellen werden von ICS in der Reihenfolge geöffnet und gelesen, in der sie definiert wurden, d.h. zuerst N1 (NOOP) und dann M1 (MSIN). Das von NOOP generierte Objekt bewirkt, dass die Benutzerprozedur MeineInitialisierung0 einmal aufgerufen wird (z.B. für spezielle Initialisierungen von Benutzermaps). In diesem Fall hätte man das Gleiche jedoch auch ohne NOOP, mit einem PRE_SOURCE_M1 Trigger, erreichen können.



Wir empfehlen die Verwendung von `NOOP`, wenn die Initialisierung nicht direkt mit einem bestimmten Inputmodul zu tun hat. Wenn es jedoch nur darum geht z.B. die Quelle `I1` zu initialisieren, dann soll auch `PRE_SOURCE_I1` für die Initialisierung der Quelle benutzt werden.

2. Beispielkonfigurationen

2.1. Einleitung

In diesem Kapitel sind einige typische ICS Konfigurationsbeispiele enthalten. Die Beispiele sollen die im letzten Kapitel vorgestellten Konzepte an einigen konkreten Problemstellungen veranschaulichen. Die Beispiele sind nach aufsteigender Komplexität geordnet und der Skriptcode ist jeweils vollständig und direkt mit ICS ausführbar.

2.2. Konfigurationen mit einem Inputmodul

Die einfachsten ICS Konfigurationen benutzen nur einen Inputmodul, sind also gar keine Schnittstellen (mit Input- und Outputmodul) im eigentlichen Sinn. Trotzdem können solche Konfigurationen sehr nützlich sein. Beispiele für Konfigurationen mit nur einem Inputmodul sind:

- Anzeigekonfigurationen, welche alle Daten (oder einen Teil der Daten) eines Inputmoduls anzeigen.
- Statistikkonfigurationen, welche eine Statistik aus den Daten des Inputmoduls erzeugen.
- Checkerkonfigurationen, welche Fehler in dem Datensatz anzeigen.

2.2.1. Anzeigekonfiguration

Der nachfolgende Skript kann den Inhalt einer beliebigen INTERLIS 1 .itf Datei in der .log Datei ausgeben. Das kann z.B. nützlich sein, um zu verstehen, welche Objekte vom ILIN Modul von einem konkreten Datensatz geliefert werden.

`\script\il2il\display.cfg`

```

=====
!
! INTERLIS 1 Display Configuration Vers. 1.5
!
!=====

!<1>
|LICENSE \license\iltoolspro.lic
|LICENSE \license\iltools.lic
|LICENSE \license\il2il.lic

!+++++
!
! user input
!

!<2>
MAP USER_INPUT1
  DIALOG => FILE ! FILE | FILES | DIRECTORY | STRING | ODBC
  MESSAGE => 'select .itf input file'
  FILE_EXISTS => TRUE
  FILE_FILTER => itf
  OPT => input

```

```

END_MAP

!+++++
!
! parameter maps for input modules
!

!<3>
MAP ILIN_PARAM
  INTERLIS_DEF => '' ! will be set from .itf file
  LOG_TABLE    => OFF
  TRACE        => OFF
  DEBUG        => OFF
  ENUM_TO_TEXT => ON
  STATISTICS   => ON
END_MAP

!+++++
!
! input sources
!

!<4>
MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
END_MAP

!+++++
!
! classification
!

!<5>
MAP INOUT
  I1 => DISPLAY_OBJECT1,IN
END_MAP

!<6>
|INCL \script\util.lib
|INCL \script\ilin.mod
|INCL \script\run1.prg

```

Erläuterungen zum Beispiel:

1. Mit der Direktive |LICENSE werden die möglichen Lizenzdateien angegeben, mit welchen der Skript ausgeführt werden kann. Werden diese Einträge vergessen, wird der Skript zur Laufzeit mit einer entsprechenden Fehlermeldung abgebrochen (no license found for ...).
2. Der Skript fragt vom Benutzer den Dateinamen der Inputdatei ab. Damit nur bestehende .itf Dateien ausgewählt werden können wird FILE_EXISTS => TRUE und FILE_FILTER => itf gesetzt. Der abgefragte Wert wird in OPT.input gespeichert (wegen OPT => input).
3. Für das Lesen der .itf Datei wird der Modul ILIN benötigt. Dieser verlangt die Übergabe der notwendigen Parameter in der Map ILIN_PARAM. Mit der Angabe INTERLIS_DEF => '' wird z.B. verlangt, dass das INTERLIS 1 Modell aus der .itf Inputdatei bestimmt werden soll.

4. In der Map INPUT_SOURCES werden die Inputquellen festgelegt. Im Fall der Anzeigekonfiguration gibt es nur eine Inputquelle I1 welche ihre Daten vom Inputmodul ILIN aus der Datei OPT.input bezieht.
5. Der Verarbeitungsablauf wird schliesslich in der Map INOUT festgelegt. In diesem Beispiel werden alle von der Inputquelle I1 gelesenen Objekte über die Prozedur DISPLAY_OBJECT1 aus util.lib in die Logdatei (normalerweise \data\temp\ics.log) ausgegeben.
6. Alle benutzten Module und Bibliotheken müssen über |INCL Direktiven eingebunden werden. In diesem Fall sind das \script\util.lib (wegen DISPLAY_OBJECT1), \script\ilin.mod (wegen INTERLIS 1 Input) und \script\run1.prg welches in jede RUN1 Konfiguration direkt oder indirekt eingebunden werden muss.

2.2.2. Statistikkonfiguration

Das folgende Script gibt eine Statistik über alle Layer einer .dxf Datei aus. Pro Layer werden die Anzahl Objekte gezählt und am Schluss der .log Datei ausgegeben.

\script\dx2il\stat.cfg

```

!=====
!
! DXF Statistic Configuration Vers. 1.0
!
!=====

!<1>
|LICENSE \license\iltoolspro.lic
|LICENSE \license\iltools.lic
|LICENSE \license\dx2il.lic

!+++++
!
! user input
!

!<2>
MAP USER_INPUT1
  DIALOG => FILE ! FILE | FILES | DIRECTORY | STRING | ODBC
  MESSAGE => 'select .dxf input file'
  FILE_EXISTS => TRUE
  FILE_FILTER => dxf
  OPT => input
END_MAP

!+++++
!
! parameter maps for input modules
!

!<3>
MAP DXFIN_PARAM
END_MAP

!+++++
!
! input sources
!

```

```

!<4>
MAP INPUT_SOURCES
  I1 => DXFIN,OPT.input
END_MAP

!+++++
!
! classification
!

!<5>
MAP INOUT
  I1 => SAVE_LAYER0
END_MAP

!<6>
|INCL \script\util.lib
|INCL \script\dxfin.mod
|INCL \script\dx2il\stat.out
|INCL \script\run1.prg

```

\script\dx2il\stat.out

```

!+++++
!
! user defined procedures
!

MAP LAYER_OBJECTS
END_MAP

!<7>
PROCEDURE SAVE_LAYER0
  IF IN.LAYER LAYER_OBJECTS IS_NULL THEN
    &LAYER_OBJECTS IN.LAYER 1 MAPINS
  ELSE
    &LAYER_OBJECTS IN.LAYER IN.LAYER LAYER_OBJECTS INC MAPINS
  END_IF
END_PROCEDURE

!<8>
PROCEDURE PRE_SOURCE_I1
  0 SET_NULL => LAYER_OBJECTS.DEFAULT
END_PROCEDURE

!<9>
PROCEDURE POST_SOURCE_I1
  DISPLAY ''
  DISPLAY 'number of objects per layer'
  DISPLAY '===== '
  DISPLAY ''
  &LAYER_OBJECTS MAPRESET
  WHILE &LAYER_OBJECTS MAPSCAN DO
    DISPLAY $,' ', $
  END_WHILE
END_PROCEDURE

```

Erläuterungen zum Beispiel:

1. Mit der Direktive `|LICENSE` werden die möglichen Lizenzdateien angegeben, mit welchen der Skript ausgeführt werden kann. Werden diese Einträge vergessen, wird der Skript zur Laufzeit mit einer entsprechenden Fehlermeldung abgebrochen (no license found for ...).
2. Der Skript fragt vom Benutzer den Dateinamen der Inputdatei ab. Damit nur bestehenden `.dxf` Dateien ausgewählt werden können wird `FILE_EXISTS => TRUE` und `FILE_FILTER => dxf` gesetzt. Der abgefragte Wert wird in `OPT.input` abgelegt (wegen `OPT => input`).
3. Für das Lesen der `.dxf` Datei wird der Modul `DXFIN` benötigt. Dieser verlangt die Übergabe der notwendigen Parameter in der Map `DXFIN_PARAM`. In diesem Beispiel müssen zwar keine konkreten Parameter gesetzt werden, die Map muss aber trotzdem vorhanden sein.
4. In der Map `INPUT_SOURCES` werden die Inputquellen festgelegt. Im Fall der Checkerkonfiguration gibt es nur eine Inputquelle `I1` welche ihre Daten vom Inputmodul `DXFIN` aus der Datei `OPT.input` bezieht.
5. Der Verarbeitungsablauf wird schliesslich in der Map `INOUT` festgelegt. In diesem Beispiel werden alle von der Inputquelle `I1` gelesenen Objekte über die Benutzerprozedur `SAVE_LAYER0` aus `stat.out` in einer Benutzermap zwischengespeichert.
6. Alle benutzten Module und Bibliotheken müssen über `|INCL` Direktiven eingebunden werden. In diesem Fall sind das `\script\util.lib` (wegen `DISPLAY_OBJECT1`), `\script\dxfin.mod` (wegen `DXF Input`), die Benutzerdefinierte `.out` Datei `\script\dx2il\stat.out` (wegen der Benutzerprozedur `SAVE_LAYER0`) und `\script\run1.prg` welches in jede `RUN1` Konfiguration direkt oder indirekt eingebunden werden muss.
7. Die Prozedur `SAVE_LAYER0` speichert die Anzahl Objekte pro Layer in der Map `LAYER_OBJECTS`.
8. Die Map `LAYER_OBJECTS` wird durch die Triggerprozedur `PRE_SOURCE_I1` am Anfang des Skripts initialisiert.
9. Am Schluss des Skripts wird der Inhalt der Map `LAYER_OBJECTS` durch die Triggerprozedur `POST_SOURCE_I1` in die `.log` Datei ausgegeben.

2.2.3. Checkerkonfiguration

Das folgende Beispiel prüft, ob in einer DXF Datei nur bestimmte Layer vorkommen. Layer welche nicht in der Map `KNOWN_LAYERS` eingetragen sind, werden als Fehler gemeldet. Nachfolgend ist die `.cfg` und die `.out` Datei dargestellt:

`\script\dx2il\check.cfg`

```

!=====
!
! DXF Check Configuration Vers. 1.0
!
!=====

!<1>
|LICENSE \license\iltoolspro.lic
|LICENSE \license\iltools.lic
|LICENSE \license\dx2il.lic

!+++++
!
! user input
!
```

```

!<2>
MAP USER_INPUT1
  DIALOG => FILE ! FILE | FILES | DIRECTORY | STRING | ODBC
  MESSAGE => 'select .dxf input file'
  FILE_EXISTS => TRUE
  FILE_FILTER => dxf
  OPT => input
END_MAP

!+++++
!
! parameter maps for input modules
!

!<3>
MAP DXFIN_PARAM
END_MAP

!<4>
MAP KNOWN_LAYERS
  ! layer => KNOWN | UNKNOWN
  01121 => KNOWN
  DEFAULT => UNKNOWN
END_MAP

!+++++
!
! input sources
!

!<5>
MAP INPUT_SOURCES
  I1 => DXFIN,OPT.input
END_MAP

!+++++
!
! classification
!

!<6>
MAP INOUT
  I1 => CHECK_LAYER0
END_MAP

| INCL \script\dxfin.mod
| INCL \script\dx2il\check.out
| INCL \script\run1.prg

```

\script\dx2il\check.out

```

!+++++
!
! user defined procedures
!

!<7>
PROCEDURE CHECK_LAYER0

```

```

IF IN.LAYER KNOWN_LAYERS = 'UNKNOWN' THEN
  ERROR 'unknown DXF layer ',IN.LAYER
END_IF
END_PROCEDURE

```

Erläuterungen zum Beispiel:

1. Mit der Direktive `|LICENSE` werden die möglichen Lizenzdateien angegeben, mit welchen der Skript ausgeführt werden kann. Werden diese Einträge vergessen, wird der Skript zur Laufzeit mit einer entsprechenden Fehlermeldung abgebrochen (no license found for ...).
2. Der Skript fragt vom Benutzer den Dateinamen der Inputdatei ab. Damit nur bestehende `.dxf` Dateien ausgewählt werden können wird `FILE_EXISTS => TRUE` und `FILE_FILTER => dxf` gesetzt. Der abgefragte Wert wird in `OPT.input` abgelegt (wegen `OPT => input`).
3. Für das Lesen der `.dxf` Datei wird der Modul `DXFIN` benötigt. Dieser verlangt die Übergabe der notwendigen Parameter in der Map `DXFIN_PARAM`. In diesem Beispiel müssen zwar keine konkreten Parameter gesetzt werden, die Map muss aber trotzdem vorhanden sein.
4. In der Map `INPUT_SOURCES` werden die Inputquellen festgelegt. Im Fall der Checkerkonfiguration gibt es nur eine Inputquelle `I1` welche ihre Daten vom Inputmodul `DXFIN` aus der Datei `OPT.input` bezieht.
5. Der Verarbeitungsablauf wird schliesslich in der Map `INOUT` festgelegt. In diesem Beispiel werden alle von der Inputquelle `I1` gelesenen Objekte über die Benutzerprozedur `CHECK_LAYER0` aus `check.out` in die Logdatei ausgegeben.
6. Alle benutzten Module und Bibliotheken müssen über `|INCL` Direktiven eingebunden werden. In diesem Fall sind das `\script\dxfin.lib` (wegen `DXF Input`), `\script\dx211\check.out` (wegen Benutzerprozedur) und `\script\run1.prg` welches in jede `RUN1` Konfiguration direkt oder indirekt eingebunden werden muss.
7. Der eigentliche Layer-Test wird in der Benutzerprozedur `CHECK_LAYER0` durchgeführt. Die Prozedur wird auf jedem von der Inputquelle `I1` gelesenen DXF Objekt aufgerufen (wegen `I1 => CHECK_LAYER0`). Weil die Prozedur keine Argumente benötigt, endet der Name der Prozedur mit der Ziffer `0`.

2.3. Konfiguration mit Input- und Outputmodul

ICS Konfigurationen mit einem Input- und einem Outputmodul sind die häufigsten ICS Konfigurationen und werden häufig einfach als Schnittstellen bezeichnet. Nachfolgend ist eine `DXF => SHP` Schnittstelle dargestellt, welche `DXF`-Block Objekte (`DXF = AutoCAD DXF Format`) in `SHP`-Point Objekte (`SHP = ESRI Shape File Format`) übersetzt:

`\script\dx2shp\dx2shp.cfg`

```

=====
!
! DXF => SHP Configuration Vers. 1.0
!
!=====
!<1>
|LICENSE \license\iltoolspro.lic
|LICENSE \license\iltools.lic
!+++++

```

```
!  
! user input  
!  
  
!<2>  
MAP USER_INPUT1  
  DIALOG => FILE ! FILE | FILES | DIRECTORY | STRING | ODBC  
  MESSAGE => 'select .dxf input file'  
  FILE_EXISTS => TRUE  
  FILE_FILTER => dxf  
  OPT => input  
END_MAP  
  
!<3>  
MAP USER_INPUT2  
  DIALOG => DIRECTORY ! FILE | FILES | DIRECTORY | STRING | ODBC  
  MESSAGE => 'select .shp output directory'  
  OPT => output  
END_MAP  
  
!+++++  
!  
! parameter maps for input modules  
!  
  
!<4>  
MAP DXFIN_PARAM  
END_MAP  
  
!+++++  
!  
! parameter maps for output modules  
!  
  
!<5>  
MAP SHPOUT_PARAM  
END_MAP  
  
!+++++  
!  
! input sources  
!  
  
!<6>  
MAP INPUT_SOURCES  
  I1 => DXFIN,OPT.input  
END_MAP  
  
!+++++  
!  
! classification  
!  
  
!<7>  
MAP INOUT  
  I1 => IN.TYPE  
  I1,BLOCK => SHPOUT_WRITE_POINT3,IN.GEOM,2D,point  
  I1,* => OFF
```

```

END_MAP

!<8>
|INCL \script\dxfin.mod
|INCL \script\shpout.mod
|INCL \script\run1.prg

```

Erläuterungen zum Beispiel:

1. Mit der Direktive `|LICENSE` werden die möglichen Lizenzdateien angegeben, mit welchen der Skript ausgeführt werden kann. Werden diese Einträge vergessen, wird der Skript zur Laufzeit mit einer entsprechenden Fehlermeldung abgebrochen (`no license found for ...`).
2. Der Skript fragt vom Benutzer den Dateinamen der Inputdatei ab. Damit nur bestehenden `.dxf` Dateien ausgewählt werden können wird `FILE_EXISTS => TRUE` und `FILE_FILTER => dxf` gesetzt. Der abgefragte Wert wird in `OPT.input` abgelegt (wegen `OPT => input`).
3. Der Skript fragt vom Benutzer das Outputverzeichnis ab. Der abgefragte Wert wird in `OPT.output` abgelegt (wegen `OPT => output`).
4. Für das Lesen der `.dxf` Datei wird der Modul `DXFIN` benötigt. Dieser verlangt die Übergabe der notwendigen Parameter in der Map `DXFIN_PARAM`. Im Fall der `DXF => SHP` Schnittstelle müssen keine speziellen Parameter gesetzt werden, die Map `DXFIN_PARAM` muss aber trotzdem definiert werden.
5. Für das Schreiben der `.shp` Datei wird der Modul `SHPOUT` benötigt. Dieser verlangt die Übergabe der notwendigen Parameter in der Map `SHPOUT_PARAM`. Im Fall der `DXF => SHP` Schnittstelle müssen keine speziellen Parameter gesetzt werden, die Map `SHPOUT_PARAM` muss aber trotzdem definiert werden.
6. In der Map `INPUT_SOURCES` werden die Inputquellen festgelegt. Im Fall der `SHP => DXF` Schnittstelle gibt es nur eine Inputquelle `I1` welche ihre Daten vom Inputmodul `DXFIN` aus der Datei `OPT.input` bezieht.
7. Der Verarbeitungsablauf wird schliesslich in der Map `INOUT` festgelegt. In diesem Beispiel werden alle von der Inputquelle `I1` gelesenen Objekte zunächst über die Komponente `TYPE` des `IN` Objekts klassifiziert. Objekte mit `IN.TYPE = 'BLOCK'` werden über die Prozedur `SHPOUT_WRITE_POINT3` des `SHPOUT` Moduls in die `SHP` Punktdatei `point` geschrieben.
8. Alle benutzten Module und Bibliotheken müssen über `|INCL` Direktiven eingebunden werden. Es sind dies: `\script\dxfin.mod` (wegen `DXF` Input), `\script\shpout.mod` (wegen `SHP` Output) und `\script\run1.prg`, welches in jede `RUN1` Konfiguration direkt oder indirekt eingebunden werden muss.

2.4. Konfiguration mit Verarbeitungsmodul

Das nächste Konfigurationsbeispiel ist eine Erweiterung des Beispiels aus ???. In diesem Beispiel sind Flächen in der `DXF` Datei nur in Form von Begrenzungslinien aber nicht als geschlossene Polygone gespeichert. In der `SHP` Outputdatei sollen aber geschlossene Polygone ausgegeben werden. Es ist also eine Umrechnung der Begrenzungslinien in Polygone notwendig. Für diese Umrechnung ist der Verarbeitungsmodul `TOPO` vorgesehen. Nachfolgend ist die vollständige Konfiguration inkl. Aufruf des Verarbeitungsmodul `TOPO` dargestellt:

```
\script\dxf2shp\dx2shp_topo.cfg
```

```

!=====
!
```

```
! DXF => SHP Configuration Vers. 1.0 !
! !
!=====!
!<1>
|LICENSE \license\iltoolspro.lic
|LICENSE \license\iltools.lic

!+++++!
!
! user input
!

!<2>
MAP USER_INPUT1
  DIALOG => FILE ! FILE | FILES | DIRECTORY | STRING | ODBC
  MESSAGE => 'select .dxf input file'
  FILE_EXISTS => TRUE
  FILE_FILTER => dxf
  OPT => input
END_MAP

!<3>
MAP USER_INPUT2
  DIALOG => DIRECTORY ! FILE | FILES | DIRECTORY | STRING | ODBC
  MESSAGE => 'select .shp output directory'
  OPT => output
END_MAP

!+++++!
!
! parameter maps for input modules
!

!<4>
MAP DXFIN_PARAM
END_MAP

!+++++!
!
! parameter maps for output modules
!

!<5>
MAP SHPOUT_PARAM
  STROKE_TOL => 0.01
END_MAP

!+++++!
!
! parameter maps for processing modules
!

!<6>
MAP TOPO_PARAM
  RESOLUTION => 0.001
  OVERLAP => 0.2
END_MAP
```

```

!+++++
!
! input sources
!

!<7>
MAP INPUT_SOURCES
  I1 => DXFIN,OPT.input
  I2 => TOPO,AREA
END_MAP

!+++++
!
! classification
!

!<8>
MAP INOUT

  I1 => IN.TYPE
  I1,BLOCK => SHPOUT_WRITE_POINT3,IN.GEOM,2D,point
  I1,POLYLINE => TOPO_WRITE_BOUNDARY1,IN.GEOM
  I1,* => OFF

  I2 => SHPOUT_WRITE_POLYGON4,IN.GEOM,2D,0.01,polygon

END_MAP

!<9>
|INCL \script\util.lib
|INCL \script\topo.mod
|INCL \script\dxfin.mod
|INCL \script\shpout.mod
|INCL \script\run1.prg

```

1. Mit der Direktive |LICENSE werden die möglichen Lizenzdateien angegeben, mit welchen der Skript ausgeführt werden kann. Werden diese Einträge vergessen, wird der Skript zur Laufzeit mit einer entsprechenden Fehlermeldung abgebrochen (no license found for ...).
2. Der Skript fragt vom Benutzer den Dateinamen der Inputdatei ab. Damit nur bestehende .dxf Dateien ausgewählt werden können wird FILE_EXISTS => TRUE und FILE_FILTER => dxf gesetzt. Der abgefragte Wert wird in OPT.input abgelegt (wegen OPT => input).
3. Der Skript fragt vom Benutzer das Outputverzeichnis ab. Der abgefragte Wert wird in OPT.output abgelegt (wegen OPT => output).
4. Für das Lesen der .dxf Datei wird der Modul DXFIN benötigt. Dieser verlangt die Übergabe der notwendigen Parameter in der Map DXFIN_PARAM. Im Fall der DXF => SHP Schnittstelle müssen keine speziellen Parameter gesetzt werden, die Map DXFIN_PARAM muss aber trotzdem definiert werden.
5. Für das Schreiben der .shp Datei wird der Modul SHPOUT benötigt. Dieser verlangt die Übergabe der notwendigen Parameter in der Map SHPOUT_PARAM.
6. Für den Verarbeitungsmodul TOPO müssen die Parameter in der Map TOPO_PARAM gesetzt werden.

7. In der Map `INPUT_SOURCES` werden die Inputquellen festgelegt. Im Fall der SHP => DXF Schnittstelle gibt es nun zwei Inputquellen I1 und I2:

I1

Liest die DXF-Objekte aus der Datei `OPT.input`.

I2

Liest die vom `TOPO` Modul aufbereiteten Polygone.

8. Der Verarbeitungsablauf der INOUT Map gliedert sich in zwei Teile:

Verarbeitung der Inputquelle I1

Die Daten der Inputquelle I1 werden einerseits als SHP-Point Objekte ausgegeben andererseits werden Polylinien im Topologiemodul `TOPO` für die weitere Verarbeitung mit I2 gespeichert.

Verarbeitung der Inputquelle I2

Die vom Topologiemodul gelieferten geschlossenen Polygone werden als SHP-Polygon Objekte ausgegeben.

9. Alle benutzten Module und Bibliotheken müssen über `|INCL` Direktiven eingebunden werden. `\script\dxfin.mod` (wegen DXF Input), `\script\shpout.mod` (wegen SHP Output) und `\script\run1.prg`, welches in jede `RUN1` Konfiguration direkt oder indirekt eingebunden werden muss.

3. Konfigurieren mit ICS

3.1. Einleitung

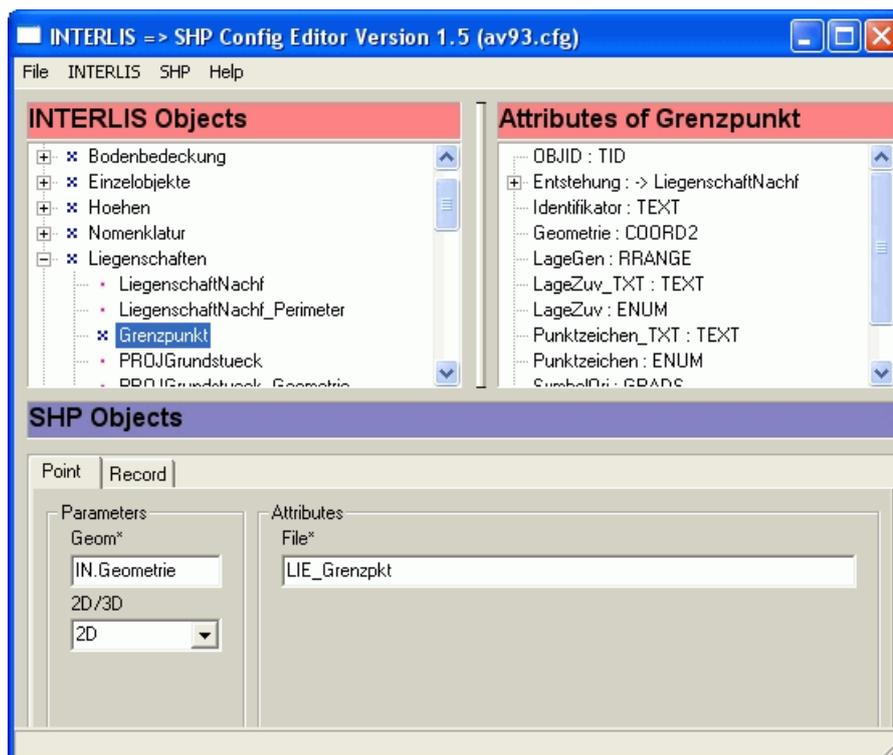
In den letzten Kapiteln wurde der generelle Aufbau von ICS und die Funktionsweise des RUN1 Algorithmus erklärt. In diesem Kapitel soll nun auf die praktische Arbeit mit den diversen ICS Werkzeugen eingegangen werden.

3.2. Verwendung von Editoren

Die iG/Script Konfigurationsdateien sind ASCII Dateien, welche mit einem beliebigen Texteditor bearbeitet werden können. Wir empfehlen die Verwendung eines Texteditors mit Anzeige- bzw. Suchfunktionen für die Zeilennummer (z.B. PFE, Ultraedit, etc.). Via die Zeilennummer können Fehler in den Skriptdateien leicht gefunden werden.

Für einige Schnittstellen existieren ausserdem spezielle Konfigurationseditoren mit einer graphischen Benutzeroberfläche (GUI), z.B. DBEDIT, DXFEDIT, SHPEDIT, etc. . Nachfolgend ist die Benutzeroberfläche von SHPEDIT dargestellt:

Abbildung 3. Konfigurationseditor SHPEDIT



Diese Editoren erzeugen ebenfalls RUN1 kompatible .cfg Dateien, welche auch von Hand (d.h. mit einem Texteditor) nachbearbeitet oder ergänzt werden können. Konsultieren Sie dazu die entsprechenden Benutzerhandbücher.

3.3. Ausführen der Konfigurationen

Für die Ausführung von iG/Script Konfigurationen kann man je nach Anwendungszweck unter vier verschiedenen Runtime Umgebungen auswählen:

ICS for Windows

Eignet sich für das Ausführen von beliebigen iG/Script Konfigurationen unter Windows. Eine detaillierte Beschreibung dieser Oberfläche ist im INTERLIS Tools Benutzerhandbuch enthalten.

ICS for Microstation

Bei ICS for Microstation handelt sich um eine direkt in das CAD Paket Microstation integrierte Version von ICS for Windows. Eine detaillierte Beschreibung dieser Oberfläche ist im INTERLIS Tools Benutzerhandbuch enthalten.

ICS Kommandozeile

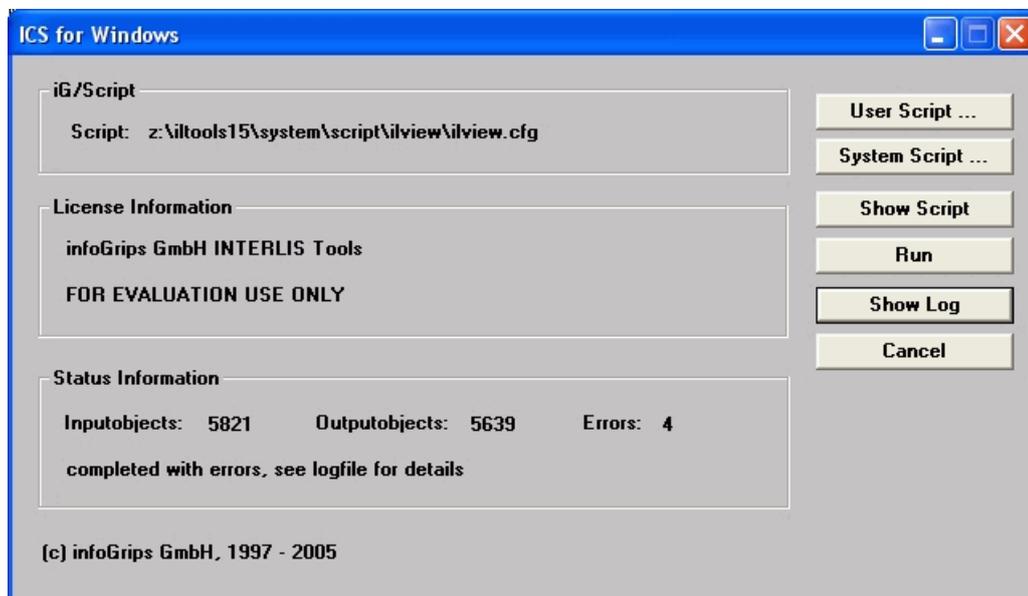
ICS Kommandozeilenversion, welche sich vorallem für die automatisierte Verarbeitung von grossen Datenbeständen oder für das Debugging eignet (s.a. nächster Abschnitt). Eine detaillierte Beschreibung dieser Oberfläche ist im INTERLIS Tools Benutzerhandbuch enthalten.

GeoShop

Der GeoShop ist ein universeller Datenserver für das Publizieren, Verteilen und Verkaufen von Geodaten im Intra- oder Internet. Der GeoShop baut auf ICS (bzw. den INTERLIS Tools) auf. iG/Script Programme welche für ICS entwickelt wurden, können auch im GeoShop ausgeführt werden. Das Produkt GeoShop ist in der separaten Dokumentation zum GeoShop beschrieben.

Als Beispiel ist nachfolgend die Benutzeroberfläche ICS for Windows dargestellt:

Abbildung 4. ICS for Windows



3.4. Hinweise und Tips

3.4.1. Fehlersuche in ICS Konfigurationen (Debugging)

Auch beim geübtesten ICS Programmierer wird einmal der Fall auftreten, dass eine Konfiguration nicht genau das macht, was er oder sie wollte, bzw. dass die Konfiguration mit einer Fehlermeldung abbricht. Bei der Fehlersuche kann man wie folgt vorgehen:

1. Zuerst sollte man die Fehlermeldung in der Logdatei (normalerweise `\data\temp\ics.log`) untersuchen. Meist findet man dort einen Hinweis auf die Zeilennummer des Problems in der Skriptdatei.
2. Mit einem Texteditor kann man dann die entsprechende Stelle im Skript untersuchen. Falls man nicht sofort sieht, wo das Problem liegt (wie z.B. bei einem Syntaxfehler), kann man mit `DISPLAY` Befehlen versuchen das Problem im Skript genauer einzukreisen. Es ist z.B. folgender Benutzerskript (`user.out`) geschrieben worden:

```
...
278 IN.TXT => OUT.TXT
...
```

Bei der Ausführung des Skripts ist dabei folgender Fehler aufgetreten:

```
*** ERROR *** ICSCPU() attribute ROOT.IN.Txt unknown user.out line 278
STACK
  EMPTY
END_STACK
PSTACK
  EMPTY
END_PSTACK
```

Dann sollte man in der Datei `user.out` vor der Zeile 278 einen `DISPLAY` Befehl wie folgt einbauen:

```
278 DISPLAY IN
279 IN.TXT => OUT.TXT
```

Bei der nächsten Ausführung des Skripts wird dann der Inhalt des `IN`-Objekts ausgegeben:

```
--- MAP IN -----
  Txt -> string(hello, world) refs:1
--- END MAP IN ---
```

Offenbar liegt also das Problem darin, dass die Komponente nicht `IN.TXT`, sondern `IN.Txt` heisst, was zum Skriptabbruch führte (in `iG/Script` ist die Gross- bzw. Kleinschrift von Komponentennamen wesentlich).



Für das Debugging eignen sich ausserdem die Methoden `ICSCPU.DISPLAY_STACK` und `ICS.DISPLAY` (s.a. `iG/Script Benutzer- und Referenzhandbuch`).

3. Das oben beschriebene Verfahren kann aber mühsam sein, wenn man jedesmal den Skript starten und dann diverse Eingaben interaktiv machen muss, bis dann der Skript endlich an der Problemstelle abbricht. Man kann daher die Testläufe automatisieren, indem man mit der ICS Kommandozeile arbeitet und den Skript mit allen notwendigen Parametern aufstartet. Der Skript fragt dann nicht mehr interaktiv nach Parametern (s.a. `INTERLIS Tools Benutzerhandbuch` für alle möglichen Parameter). Nachfolgend ist ein Beispiel mit der ICS Kommandozeile angegeben:

```
ics.exe -script \il2dxf\Grunddatensatz.cfg
        -input test.itf
        -output test.dxf
```

Bei sehr vielen Parametern kann man ausserdem eine Optionendatei (hier: test.opt) wie folgt anlegen:

```
script \il2dxf\Grunddatensatz.cfg
input test.itf
output test.dxf
log c:\test.log
...
etc.
```

Die Optionendatei test.opt kann man dann wie folgt anwenden:

```
ics.exe -opt test.opt
```

3.4.2. Batchaufruf von ICS Konfigurationen

Muss man sehr viele Dateien verarbeiten, ist es sinnvoll, eine Batchdatei anzulegen.

Beispiel 5. Batchdatei translate.bat

```
ICS_DIR\system\bin\ics.exe -script \script\il2dxf\Grunddatensatz.cfg \
                          -input test1.itf -output test1.dxf
ICS_DIR\system\bin\ics.exe -script \script\il2dxf\Grunddatensatz.cfg \
                          -input test2.itf -output test2.dxf
ICS_DIR\system\bin\ics.exe -script \script\il2dxf\Grunddatensatz.cfg \
                          -input test3.itf -output test3.dxf
...
etc.
...
ICS_DIR\system\bin\ics.exe -script \script\il2dxf\Grunddatensatz.cfg \
                          -input testn.itf -output testn.dxf
```

Die Batchdatei translate.bat kann man dann von der MSDOS Kommandozeile wie folgt aufrufen:

```
$ translate.bat
```

Für eine noch höhere Automatisierung kann man den Batchaufruf ausserdem in einen Systemtask (Geplante Tasks in der Windows-Systemsteuerung) einbetten, welcher dann zu einem bestimmten Zeitpunkt jeweils wieder ausgeführt wird. Konsultieren Sie dazu die entsprechenden Dokumentationen der Windows Betriebssysteme.

Schliesslich steht mit dem GeoShop ein Produkt zur Verfügung mit dem man jeden ICS Script ins Intra- oder Internet stellen kann. Mögliche GeoShop Anwendungen sind:

- Automatisierte Lieferung von Daten an Dritte via Dateisystem, FTP oder E-Mail.
- Automatische Verrechnung der Datenbezüge.
- Visualisierung der Daten im Internet- bzw. Intranet.
- etc.

Das Produkt GeoShop ist in separaten Benutzerhandbüchern beschrieben.

A. Input Module

1. Einleitung

In diesem Anhang sind alle Input Module und ihre Prozeduren bzw. Methoden beschrieben, welche zusammen mit dem RUN1Algorithmus benutzt werden können.

2. Modul ARCGISIN - ESRI Geodatabase lesen

2.1. Allgemeines

Mit dem Modul können Objekte aus einer ESRI-Geodatabase gelesen werden. Unterstützt werden:

- ESRI SDE Geodatabase (Oracle)
- ESRI Personal Geodatabase (ACCESS)
- ESRI File Geodatabase

Der Modul wird mit:

```
|INCL \script\arcgisin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

2.2. ESRI Lizenz

Der Modul verwendet das ESRI ArcObjects API. Für die Anwendung des Modules ist deshalb im Minimum eine ESRI Lizenz ArcGIS Engine Runtime notwendig.

2.3. Parametermap ARCGISIN_PARAM

Folgende Parameter können in der Map ARCGISIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
FENCE	o	string. Fence Definition. Es werden nur Objekte gelesen, die innerhalb des Fences liegen. Objekte ohne Geometrien werden vollständig gelesen. Der Fence kann als Rechteck in der Form minx/miny,maxx/maxy oder als geschlossenes Polygon in der Form x1/y1,x2/y2, ... xi/yi, ... xn/yn definiert werden. Der Fence kann auch durch die Option OPT.fence gesetzt werden.
FENCE_MODE	o	OFF, INSIDE, OVERLAP. Definiert den Modus für den Fence. OFF: Der Fence wird nicht berücksichtigt. INSIDE : Es werden nur Objekte gelesen, die vollständig innerhalb des Fences liegen.OVERLAP : Es werden nur Objekte gelesen, die innerhalb des Fences liegen oder den Fence überlappen.
READ_M	o	ON oder OFF. Falls ON gesetzt wird, werden die Measurement-Werte als Z-Koordinaten gelesen, falls vorhanden. Eventuelle Z-

		Koordinaten werden nicht gelesen. Measurement-Werte können in folgenden Geometrie Typen definiert werden: POINT, MULTIPOINT, POLYLINE, POLYGON.
--	--	---

2.4. ArcGIS SDE Connect

Der Connect zu einer ArcGIS SDE Datenbank wird unterschieden zwischen einem

- Direct Connect (ab 10.1 Standard)

und einem

- SDE Service Connect (bis 10.0 Standard)

Bis und mit SDE Version 10.0 war der Connect über einen SDE Service der Standard. Ab SDE Version 10.1 ist Direct Connect der Standard. Direct Connect benötigt keinen SDE Service, sondern verwendet die Möglichkeiten der Datenbank für einen Connect.

Bis SDE Version 10.0 konnte der SDE Service über das Postinstallations-Programm von SDE interaktiv eingerichtet werden. Ab SDE Version 10.1 muss der SDE Service manuell eingerichtet werden.

Entsprechend der Verbindung über Direct Connect oder SDE Service sind die Connect-Parameter etwas anders. Nachfolgend werden Beispiele für die Verbindung zweier geläufigen Datenbanken aufgeführt.

Datenbank	Connect	Connect Parameter Interaktiv	Connect Parameter Batch OPT.input/OPT.output
Oracle	Direct Connect	Server: '' Leer. Instance: sde:<Oracle-Client-Value> sde: mit Oracle-Client-Version je nach Client Version: Oracle (für 8i), Oracle9i, Oracle10g, Oracle11g Database: <Database> (Optional) Datenbank des Users. User: <User> Datenbank User Password: <Password>@<Oracle-Service> Password des Users und Oracle Service definiert für Client Version: <Version> (Optional) Leer oder Default sde.DEFAULT.	Parameter ,<Instance>,<Database>,<User>,<Password>@<Oracle-Service>,<Version> Beispiel ,sde:Oracle11g,,test,info-grips@ORCL,
	SDE Service	Server: <Server> Server mit SDE Service Instance: <SDE Service Port> Port des SDE Services	Parameter <Server>,<Instance>,<Database>,<User>,<Password>,<Version>

		<p>Database: <Database> (Optional) Datenbank des Users.</p> <p>User: <User> Datenbank User</p> <p>Password: <Password> Password des Users und Oracle Service definiert für Client</p> <p>Version: <Version> (Optional) Leer oder Default sde.DEFAULT.</p>	<p>Beispiel Server,5151,,test,infogrips,</p>
SQL Server	Direct Connect	<p>Server: <Server> Server mit SDE Datenbank.</p> <p>Instance: sde:sqlserver:<SQL Server Instance> sde:sqlserver mit SQL Server Instance</p> <p>Database: <Database> (Optional) Datenbank des Users.</p> <p>User: <User> Datenbank User</p> <p>Password: <Password> Password des Users</p> <p>Version: <Version> (Optional) Leer oder Default dbo.DEFAULT.</p>	<p>Parameter <Server>,<Instance>,<Database>,<User>,<Password>,<Version></p> <p>Beispiel server,sde:sqlserver:server\SQLEXPRESS,,test,infogrips,</p>
	SDE Service	<p>Server: <Server> Server mit SDE Service</p> <p>Instance: <SDE Service Port> Port des SDE Services</p> <p>Database: <Database> (Optional) Datenbank des Users.</p> <p>User: <User> Datenbank User</p> <p>Password: <Password> Password des Users</p> <p>Version: <Version> (Optional) Leer oder Default dbo.DEFAULT.</p>	<p>Parameter <Server>,<Instance>,<Database>,<User>,<Password>,<Version></p> <p>Beispiel server,5151,,test,infogrips,</p>

2.5. Objektmodell

Der Modul liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.TABLE(s)	r	Tablename des IN Objekts.
IN.GDB_DATASET(s)	r	Enthält den Geodatabase Dataset-Namen.
IN.<Attribute>(*)	o	Pro Attribut in der Geodatabase eine Komponente mit dem Attributnamen und dem Wert.
IN.<Geom-Attribute>(g li)	o	Falls es sich beim Attribut um eine Geometrie handelt, so enthält diese Komponente die Geometrie. Sind mehrere Geometrien vorhanden, so enthält die Komponente eine Liste von Geometrien.

Alle weiteren Komponenten sind abhängig von der Tabellen-Definition in der Datenbank.

2.6. Exportierte Prozeduren und Methoden

Der Modul ARCGISIN stellt folgende Prozeduren und Methoden zur Verfügung.

Prozedur **ARCGISIN_OPEN [s input][]**
Beschreibung Öffnet eine bestehende Datenbank. Für eine Personal Geodatabase muss in input die Access-Datenbank *.mdb gesetzt werden. Für eine File Geodatabase muss in input die File-Datenbank *.gdb gesetzt werden. Für eine SDE Geodatabase muss in input die SDE connection in der Form <server>, <instance>, <database>, <user>, <password>, <version> gesetzt werden. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel 'c:\data\arcgis.mdb' ARCGISIN_OPEN

Prozedur **ARCGISIN_READ_OBJECT [][b state]**
Beschreibung Liest das nächste IN-Objekt aus der geöffneten Datenbank-Tabelle. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel ARCGISIN_READ_OBJECT [TRUE]

Prozedur **ARCGISIN_CLOSE [][]**
Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel ARCGISIN_CLOSE

2.7. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von ARCGISIN.mod
! gelesenen Objekte in der .log Datei an.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG            => ARCGIS
  MESSAGE          => 'Enter SDE or Database File'
  FILE_FILTER      => mdb;gdb
  FILE_EXISTS      => TRUE
  OPT               => input
END_MAP
```

```
MAP ARCGISIN_PARAM
```

```

STATISTICS      => ON
END_MAP

MAP INPUT_SOURCES
  I1 => ARCGISIN,*
END_MAP

MAP INOUT
  I1 => DISPLAY_OBJECT1,IN
END_MAP

|INCL \script\ARCGISIN.mod
|INCL \script\run1.prg

```

2.8. Bestehende Konfigurationen IL2GDB/IL2SDE oder GDB2IL/SDE2IL nach IL2ARCGIS/ARCGIS2IL migrieren

Die Module ARCGISIN/ARCGISOUT lösen die Module GDBIN/GDBOUT und SDEIN/SDEOUT ab. Bestehende Konfigurationen IL2GDB/GDB2IL und IL2SDE/SDE2IL sind deshalb durch Konfiguration IL2ARCGIS/ARCGIS2il abzulösen. Für dieser Migration steht folgendes Script zur Verfügung.

```
ILTOOLS\system\script\il2gdb\CFG_GDB2ARCGIS.cfg
```

Das Script verlangt als Input eine IL2GDB/GDB2IL oder IL2SDE/SDE2IL Konfiguration und schreibt als Output eine analoge IL2ARCGIS/ARCGIS2IL Konfiguration.

3. Modul DBIN - ODBC-Datenbank lesen

3.1. Allgemeines

Mit dem Skriptmodul können Objekte aus einer ODBC-Datenbank (z.B. MS-Access, Oracle) gelesen werden.

Der Modul wird mit:

```
|INCL \script\dbin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

3.2. Abhängigkeiten von anderen Modulen

Der Modul importiert die Klasse DB. Es stehen daher auch alle Methoden der Klasse DB zur Verfügung (s.a. iG/Script Benutzer- und Referenzhandbuch).

3.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.

USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

3.4. Parametermap DBIN_PARAM

Folgende Parameter können in der Map DBIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
SELECT_PARSE	o	ON oder OFF, Default = ON. Definiert, ob ein eventuelles SQL-select Statement geparkt werden soll. Als Argument für für das Modul kann ein SQL-select Statement definiert werden. Als Default wird dieses select-Statement von geparkt, um spezielle Attribute zu erkennen, die vom ODBC-Treiber nicht unterstützt werden und speziell gelesen werden müssen. Manchmal können select-Statements (komplexe Joins) nicht erfolgreich geparkt werden. In solchen Fällen ist dieser Parameter auf OFF zu schalten. Sie müssen besorgt sein, dass das select-Statement von ODBC verarbeitet werden kann.

3.5. Objektmodell

Der Modul liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.TABLE(s)	r	Tablename des IN Objekts.

Alle weiteren Komponenten sind abhängig von der Tabellen-Definition in der Datenbank.

3.6. Datenbankmodell

Die Methode DB.GET_INFO liefert das Datenmodell der Datenbank in einer Objektstruktur. Auf die einzelnen Komponenten dieser Objektstruktur kann zugegriffen werden.

Anwendung der Methode DB.GET_INFO:

```
MAP DB_MODEL
END_MAP

&DB_MODEL DB.GET_INFO
```

Objektstruktur Datenmodell:

```
DB_MODEL (m)
  DB_NAME      -> (s) <Name>
  DB_USER      -> (s) <DB-User>
  DB_PASSWORD  -> (s) <DB-Password>
```

```

PRODUKT      -> (s) <Produkt>
VERSION      -> (s) <Version>
TABLES       -> (m)
:
<Tablename> -> (m)
  TABLENAME      -> (s) <Tablename>
  TABLEQUALIFIER -> (s) <Tabelqualifier>
  TABLEOWNER     -> (s) <Tableowner>
  TABLETYPE      -> (s) <Tabletype>
  COLUMNS        -> (m)
:
  <Columnname> -> (m)
    NAME          -> (s) <Name>
    TYPENAME      -> (s) <Typename>
    DATATYPE      -> (i) <Datatype>
    LENGTH        -> (i) <Length>
    PRECISION     -> (i) <Precision>
    SCALE         -> (i) <Scale>
    RADIX         -> (i) <Radix>
    NULLABLE     -> (b) <Nullable>
    REMARKS      -> (s) <Remarks>

```

Zugriffsbeispiele Objektstruktur Datenmodell:

```

! Display User,Product,Version
DISPLAY DB_MODEL

! Display Tables
DISPLAY DB_MODEL.TABLES

! Display Table Category
DISPLAY DB_MODEL.TABLES.CATEGORY

! Display Columns of Table Category
DISPLAY DB_MODEL.TABLES.CATEGORY.COLUMNS

! Display Column MSLINK of Table Category
DISPLAY DB_MODEL.TABLES.CATEGORY.COLUMNS.MSLINK

! Display Typename of Column MSLINK of Table Category
DISPLAY DB_MODEL.TABLES.CATEGORY.COLUMNS.MSLINK.TYPENAME

```

3.7. Exportierte Prozeduren und Methoden

Prozedur DBIN_OPEN [s input][[]]

Beschreibung Öffnet eine Datenbank definiert in DB_PARAM.SOURCE und liest Objekte von der Datenbank in Abhängigkeit von <input>. Für <input> können folgende Werte verwendet werden.

*

Liest alle Tabellen der Datenbank.

<tablename>

Liest die Records der Tabelle <tablename>.

<tablename>,<sql-select-statement>

Liest die Records der Tabelle <tablename> entsprechend dem SQL-select-Statement. Beinhaltet das SQL-select-Statement Hochkommas für Strings so sind die Hochkommas ' durch Anführungszeichen " zu ersetzen.

Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
'CATEGORY,select * from CATEGORY where CNAME like "av%"' DBIN_OPEN
```

```
MAP INPUT_SOURCES
```

```
    I1 => DBIN,CATEGORY,'select * from CATEGORY where CNAME like "av%"'
END_MAP
```

Prozedur

DBIN_READ_OBJECT [[**b state**]

Beschreibung

Liest das nächste IN-Objekt aus der geöffneten Datenbank-Tabelle. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
DBIN_READ_OBJECT [TRUE]
```

Prozedur

DBIN_CLOSE [[]]

Beschreibung

Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
DBIN_CLOSE
```

3.8. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von DBIN
! gelesenen Objekte in der .log Datei an.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
```

```
    DIALOG      => ODBC
```

```
    OPT         => input
```

```
END_MAP
```

```
MAP DB_PARAM
```

```
END_MAP
```

```
MAP DBIN_PARAM
```

```
    STATISTICS  => ON
```

```
END_MAP
```

```
MAP INPUT_SOURCES
```

```
    I1 => DBIN, *
```

```
END_MAP
```

```
MAP INOUT
```

```
    I1 => DISPLAY_OBJECT1, IN
```

```
END_MAP
```

```
|INCL \script\dbin.mod
```

```
|INCL \script\run1.prg
```

4. Modul DGNIN - Bentley Microstation DGN lesen

4.1. Allgemeines

Mit dem Modul können Objekte direkt aus Microstation Designfiles gelesen werden.



Das analoge Modul MSIN benötigt zum Lesen von Designfiles das Produkt Microstation.

Der Modul wird mit:

```
| INCL \script\dgnin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

4.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

4.3. Parametermap DGNIN_PARAM

Folgende Parameter können in der Map DGNIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
REFERENCEFI- LES_SCAN	o	ON oder OFF. Default OFF. Sollen die Elemente der Referenzfiles gelesen werden.
RSC_DIR	o	STRING. Definiert den Pfad mit den Microstation-Resource-Files, wie zum Beispiel die Fonts. Wird benötigt um zusätzliche Informationen zu den Objekten verarbeiten zu können. Beispiel Fontname bei Texten.
DEBUG	o	ON oder OFF. Default OFF. Debugmodus ein oder aus.
STATISTICS	o	ON oder OFF. Default OFF. Statistik anzeigen.

4.4. Map für Textjustierung

Das Modul verfügt über eine Map, die die Abbildung der Microstation-Textjustification zur INTERLIS-Textjustierung - HALIGNMENT,VALIGNMENT beinhaltet. Diese Map sieht wie folgt aus.

```
MAP DGNIN_JUST_TO_ALI
  LT => 0,1
  LC => 0,2
  LB => 0,3
  CT => 1,1
  CC => 1,2
  CB => 1,3
  RT => 2,1
  RC => 2,2
  RB => 2,3
  DEFAULT => 1,2
END_MAP
```

Diese Map kann in einer Konfiguration wie folgt angewendet werden.

```
IN.JUST DGNIN_JUST_TO_ALI EXTRLS
=> OUT.NamHALi
=> OUT.NamVALi
```

4.5. Objektmodell

Allgemeine Komponenten in jedem IN-Objekt

Komponente	req/opt	Beschreibung
IN.TYPE(i)	r	Microstation Type des Objekts.
IN.LEVEL(i)	r	Microstation Level-Nummer Objekts.
IN.LEVELNAME(s)	r	Microstation Level-Name des Objekts.
IN.COLOR(i)	r	Microstation Color des Objekts.
IN.WEIGHT(i)	r	Microstation Weight des Objekts.
IN.STYLE(i)	r	Microstation Style-Nummer des Objekts.
IN.STYLENAME(s)	r	Microstation Style-Name des Objekts.
IN.GROUP(i)	r	Microstation Graphic Group-Nummer des Objekts.
IN.FILENR(i)	r	Microstation Filenummer des Inputfiles des Objekts.
IN.FILEPOS(i)	r	Microstation Fileposition des Objekts. (Nicht implementiert)
IN.DIR(s)	r	Directory des Inputfiles des Objekts.
IN.FILE(s)	r	Filename des Inputfiles des Objekts.

Zusätzliche Komponententypen für LINE (3), LINESTRING (4), CURVE (11), ARC (16), COMPLEX_LINESTRING (12)

Komponente	req/opt	Beschreibung
IN.GEOM(l)	r	Linien-Geometrie des Objekts.

Zusätzliche Komponenten für SHAPE (6), COMPLEX_SHAPE (14), ELLIPSE (15)

Komponente	req/opt	Beschreibung
IN.GEOM(a)	r	Flächen-Geometrie des Objekts.

Zusätzliche Komponenten für TEXT (17), SYMBOL (17)

Komponente	req/opt	Beschreibung
IN.GEOM(p)	r	Punkt-Geometrie des Objekts.
IN.TXT(s)	r	Text des Objekts.
IN.TW(r)	r	Textbreite des Objekts. (Microstation: tw=).
IN.TH(r)	r	Texthöhe des Objekts. (Microstation: th=).
IN.ROT(r)	r	Rotation des Objekts.
IN.FONT(i)	r	Font des Objekts. (Microstation: ft=).
IN.JUST(s)	r	Textjustierung des Objekts. (Horizontal: L: Left, C:Center, R:Right; Vertikal: T:Top, C:Center, B:Bottom; Beispiel LC).
IN.XGEOM(li)	o	Liste von Geometrien des Objekts falls es ein Symbol ist, das heisst wenn der Font ein Symbolfont ist. (Noch nicht implementiert).

Zusätzliche Komponenten für TEXT_NODE (7)

Komponente	req/opt	Beschreibung
IN.GEOM(p)	r	Punkt-Geometrie des Objekts.
IN.TXT(s)	r	Text 1. Zeile des Objekts.
IN.TXT2(s)	0	Text 2. Zeile des Objekts.
IN.TXT<n>(s)	o	Text n. Zeile des Objekts.
IN.TW(r)	r	Textbreite des Objekts. (Microstation: tw=).
IN.TH(r)	r	Texthöhe des Objekts. (Microstation: th=).
IN.ROT(r)	r	Rotation des Objekts.
IN.FONT(i)	r	Font des Objekts. (Microstation: ft=).
IN.JUST(s)	r	Textjustierung des Objekts. (Horizontal: L: Left, C:Center, R:Right; Vertikal: T:Top, C:Center, B:Bottom; Beispiel LC).
IN.LS(r)	r	Linespacing des Objekts. (Microstation: ls=).

Zusätzliche Komponenten für CELL (2), SHARED_CELL (35)

Komponente	req/opt	Beschreibung
IN.GEOM(p)	r	Punkt-Geometrie des Objekts.
IN.CELL(s)	r	Zellname des Objekts.
IN.ROT(r)	r	Rotation des Objekts.
IN.SCALE(r)	r	Skalierung des Objekts.
IN.XGEOM(li)	r	Liste von Geometrien des Objekts.

4.6. Exportierte Prozeduren und Methoden

Prozedur	DGNIN_OPEN ! [s input][]
Beschreibung	Öffnet das Designfile <input> und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>' ' DGNIN_OPEN</pre>
Prozedur	DGNIN_READ_OBJECT ! [][b state]
Beschreibung	Liest das nächste IN-Objekt aus dem aktuellen Designfile. Falls kein Objekt mehr gelesen werden kann, wird der Status FALSE geliefert. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>DGNIN_READ_OBJECT [TRUE]</pre>
Prozedur	DGNIN_CLOSE ! [][]
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>DGNIN_CLOSE</pre>

4.7. Skriptbeispiel

Beispiel ohne Datenbankbindung.

```

! Diese ICS Konfiguration zeigt alle von dgnin.mod
! gelesenen Objekte in der .log Datei an.
! Eine eventuelle Datenbankbindung wird nicht berücksichtigt.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE
  MESSAGE => 'Enter .dgn Input File'
  FILE_FILTER => dgn
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP DGNIN_PARAM
  STATISTICS      => ON
END_MAP

MAP INPUT_SOURCES
  I1 => DGNIN,OPT.input
END_MAP

MAP INOUT
  I1 => DISPLAY_OBJECT1,IN
END_MAP

|INCL \script\dgnin.mod
|INCL \script\run1.prg

```

5. Modul DXFIN - AutoCAD DXF lesen

5.1. Allgemeines

Mit dem Modul DXFIN können Objekte aus einer AutoCAD DXF Datei gelesen werden.

Der Modul wird mit:

```
|INCL \script\dxfin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

5.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

5.3. Parametermap DXFIN_PARAM

Folgende Parameter können in der Map DXFIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
ERROR_OBJECTS	o	Unbekannte Objekte als Fehler melden (ON oder OFF).
ATTRIB_AS_TEXT	o	Attribute als Text liefern.

HEADEROBJECTS	o	Lesen und Display von Headerobjects, z.B. Blockdefinitionen (ON oder OFF).
HEADERINFO	o	Lesen und Display von Headerinformationen (ON oder OFF).
ATTRIB_AS_TEXT	o	EXTENDED Entities als TEXT liefern (ON oder OFF).
ATTRIB_SCAN	o	EXTENDED Entities lesen (ON oder OFF).
MTEXT_AS_TEXT	o	MTEXT als TEXT liefern (ON oder OFF).
DEBUG	r	DEBUG Modus ein- oder ausschalten (ON oder OFF).
STATISTICS	r	ON oder OFF, Default = OFF. Statistik anzeigen.

5.4. Objektmodell

Allgemeine Komponenten für jedes IN-Objekt

Komponente	req/opt	Beschreibung
IN.TYPE(s)	r	Objekttyp. Mögliche Werte sind: POINT, BLOCK, SHAPE, CIRCLE, TEXT, LINE, POLYLINE, LWPOLYLINE, ELLIPSE, ARC.
IN.LINE(i)	r	Zeilennummer in der Inputdatei.
IN.OBJID(s)	r	Eindeutige Objektidentifikation.
IN.LAYER(s)	r	DXF-Layer.
IN.LTYPE(i)	r	DXF-Linientyp.
IN.COLOR(i)	r	DXF-Farbe (0 .. 255).
IN.THICKNESS	r	DXF-Thickness.

Zusätzliche Komponenten für DXF-Extended Data

Das IN-Objekt kann ausserdem mehrere Komponenten als DXF-Extended Data aufweisen. Die erste DXF-Extended Data Komponente hat den Namen IN.EXTENDED1, alle weiteren erhalten im Namen als Suffix einen fortlaufenden Index: IN.EXTENDED2, IN.EXTENDED3,etc.

Komponente	req/opt	Beschreibung
IN.EXTENDED1(m)	o	1. DXF-Extended Data.
IN.EXTENDED2(m)	o	2. DXF-Extended Data.
IN.EXTENDEDn(m)	o	n. DXF-Extended Data.

Jede DXF-Extended Data Map ist wie folgt aufgebaut:

Komponente	req/opt	Beschreibung
APPLICATION(s)	r	1. DXF-Extended Data.
LAYER(s)	r	2. DXF-Extended Data.
NUMBER(s)	r	n. DXF-Extended Data.
ATTRIBU-TE1(1,r,s)	o	Attributwert 1. Attribut.
ATTRIBU-TE1(2,r,s)	o	Attributwert 2. Attribut.
ATTRIBU-TE1(n,r,s)	o	Attributwert n. Attribut.

Zusätzliche Komponenten für IN.TYPE = 'POINT'

Komponente	req/opt	Beschreibung
IN.GEOM(p)	r	Punktposition.

Zusätzliche Komponenten für IN.TYPE = 'BLOCK'

Komponente	req/opt	Beschreibung
IN.GEOM(p)	r	Blockposition.
IN.BLOCK(s)	r	Blockname.
IN.SCALE(r)	r	Skalierung.
IN.ROT(r)	r	Rotationswinkel in Altgrad.

Zusätzliche Komponenten für IN.TYPE = 'SHAPE'

Komponente	req/opt	Beschreibung
IN.GEOM(p)	r	Shapeposition.
IN.SHAPE(s)	r	Shapename.
IN.SIZE(r)	r	Grösse des Symbols.
IN.ROT(r)	r	Rotationswinkel in Altgrad.

Zusätzliche Komponenten für IN.TYPE = 'CIRCLE'

Komponente	req/opt	Beschreibung
IN.GEOM(p)	r	Zentrumsposition.
IN.RADIUS(s)	r	Kreisradius.

Zusätzliche Komponenten für IN.TYPE = 'TEXT'

Komponente	req/opt	Beschreibung
IN.TXT(s)	r	Textinhalt.
IN.GEOM(p)	r	Textposition.
IN.HEIGHT(r)	r	Texthöhe.
IN.STYLE(s)	r	Textfont.
IN.JUST(s)	r	Justierung. Mögliche Werte sind: LEFT, CENTER, RIGHT, MIDDLE.
IN.HJUST(i)	r	Horizontale Justierung (DXF-Gruppencode 72).
IN.VJUST(i)	r	Vertikale Justierung (DXF-Gruppencode 73).
IN.SLANT(r)	r	Textneigungswinkel in Altgrad.
IN.XSCALE(r)	r	Textskalierung entlang der X-Achse.
IN.ROT(r)	r	Rotationswinkel in Altgrad.

Zusätzliche Komponenten für IN.TYPE = 'MTEXT'

Komponente	req/opt	Beschreibung
IN.TXT(s)	r	Textinhalt. Zeilenumbrüche sind im Text codiert.
IN.GEOM(p)	r	Textposition.
IN.HEIGHT(r)	r	Texthöhe.
IN.STYLE(s)	r	Textfont.

IN.JUST(s)	r	Justierung. Mögliche Werte sind: LEFT, CENTER, RIGHT, MIDDLE.
IN.HJUST(i)	r	Horizontale Justierung (DXF-Gruppencode 72).
IN.VJUST(i)	r	Vertikale Justierung (DXF-Gruppencode 73).
IN.SLANT(r)	r	Textneigungswinkel in Altgrad.
IN.XSCALE(r)	r	Textskalierung entlang der X-Achse.
IN.ROT(r)	r	Rotationswinkel in Altgrad.

Zusätzliche Komponenten für IN.TYPE = 'LINE'

Komponente	req/opt	Beschreibung
IN.GEOM(l)	r	Liniengeometrie mit genau zwei Punkten.

Zusätzliche Komponenten für IN.TYPE = 'POLYLINE'

Komponente	req/opt	Beschreibung
IN.GEOM(l, a)	r	Linien- oder Flächengeometrie.
IN.FLAG(i)	r	DXF-Polyline Flag. Flag = 0 Polyline ist offen. Flag = 1 Polyline ist geschlossen.

Zusätzliche Komponenten für IN.TYPE = 'LWPOLYLINE'

Komponente	req/opt	Beschreibung
IN.GEOM(l, a)	r	Linien- oder Flächengeometrie.
IN.FLAG(i)	r	DXF-Polyline Flag. Flag = 0 Polyline ist offen. Flag = 1 Polyline ist geschlossen.

Zusätzliche Komponenten für IN.TYPE = 'ELLIPSE'

Komponente	req/opt	Beschreibung
IN.GEOM(p)	r	Zentrumsposition.
IN.POINT(p)	r	Endpunkt auf der Hauptachse, relativ zur Zentrumsposition.
IN.RATIO(r)	r	Ratio Nebenachse zur Hauptachse.
IN.START_ANGLE(r)	r	Startwinkel der Ellipse, 0.0 für eine geschlossene Ellipse.
IN.END_ANGLE(r)	r	Endwinkel der Ellipse 2*PI für eine geschlossene Ellipse.

Zusätzliche Komponenten für IN.TYPE = 'ARC'

Komponente	req/opt	Beschreibung
IN.GEOM(l)	r	Kreisbogengeometrie.
IN.CENTER(p)	r	Kreisradius.

IN.RADIUS(s)	r	Kreisradius.
IN.START_ANGLE(r)	r	Startwinkel des Kreisbogen.
IN.END_ANGLE(r)	r	Endwinkel des Kreisbogen.

5.5. Exportierte Prozeduren und Methoden

Prozedur DXFIN_OPEN ! [s input][]

Beschreibung Öffnet eine bestehende DXF Datei <input> und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel 'test.dxf' DXFIN_OPEN

Prozedur DXFIN_READ_OBJECT ![][b state]

Beschreibung Liest das nächste IN-Objekt aus der geöffneten DXF Datei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel DXFIN_READ_OBJECT [TRUE]

Prozedur DXFIN_CLOSE ! [[]]

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel DXFIN_CLOSE

5.6. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von dxfin.mod
! gelesenen Objekte in der .log Datei an.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG => FILE ! FILE | FILES | STRING | ODBC
  MESSAGE => 'Enter .dxf Input File'
  FILE_FILTER => dxf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP
```

```
MAP DXFIN_PARAM
  TRACE      => OFF
  STATISTICS => ON
END_MAP
```

```
MAP INPUT_SOURCES
  I1 => DXFIN,OPT.input
END_MAP
```

```
MAP INOUT
  I1 => DISPLAY_OBJECT1, IN
END_MAP
```

```
| INCL \script\dxfin.mod
| INCL \script\run1.prg
```

6. Modul FILEGDBIN - ESRI File-Geodatabase lesen

6.1. Allgemeines

Mit dem Modul können Objekte aus einer ESRI File-Geodatabase gelesen werden.

Der Modul wird mit:

```
| INCL \script\filegdbin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

6.2. ESRI Lizenz

Der Module benötigt **keine** ESRI Lizenz. Der Modul verwendet das ESRI File Geodatabase API.

rcGIS Engine Runtime Lizenz.

6.3. Parametermap FILEGDBIN_PARAM

Folgende Parameter können in der Map FILEGDBIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
FENCE	o	string. Fence Definition. Es werden nur Objekte gelesen, die innerhalb des Fences liegen. Objekte ohne Geometrien werden vollständig gelesen. Der Fence kann als Rechteck in der Form minx/miny,maxx/maxy definiert werden. Der Fence kann auch durch die Option OPT.fence gesetzt werden.
FENCE_MODE ! not implemented yet	o	OFF, INSIDE, OVERLAP. Definiert den Modus für den Fence. OFF: Der Fence wird nicht berücksichtigt. INSIDE : Es werden nur Objekte gelesen, die vollständig innerhalb des Fences liegen.OVERLAP : Es werden nur Objekte gelesen, die innerhalb des Fences liegen oder den Fence überlappen.
READ_M	o	ON oder OFF. Falls ON gesetzt wird, werden die Measurement-Werte als Z-Koordinaten gelesen, falls vorhanden. Eventuelle Z-Koordinaten werden nicht gelesen. Mesurement-Werte können in fogenden Geometrie Typen definiert werden: POINT, MULTIPOINT, POLYLINE, POLYGON.

6.4. Objektmodell

Der Modul liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
------------	---------	--------------

IN.TABLE(s)	r	Tablename des IN Objekts.
IN.GDB_DATASET(s)	r	Enthält den Geodatabase Dataset-Namen.
IN.<Attribute>(*)	o	Pro Attribut in der Geodatabase eine Komponente mit dem Attributnamen und dem Wert.
IN.<Geom-Attribute>(g li)	o	Falls es sich beim Attribut um eine Geometrie handelt, so enthält diese Komponente die Geometrie. Sind mehrere Geometrien vorhanden, so enthält die Komponente eine Liste von Geometrien.

Alle weiteren Komponenten sind abhängig von der Tabellen-Definition in der Datenbank.

6.5. Exportierte Prozeduren und Methoden

Der Modul FILEGDBIN stellt folgende Prozeduren und Methoden zur Verfügung.

Prozedur	FILEGDBIN_OPEN [s input][[]]
Beschreibung	Öffnet eine bestehende Datenbank. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'c:\data\filegdb.gdb' FILEGDBIN_OPEN</code>
Prozedur	FILEGDBIN_READ_OBJECT [][[b state]
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten Datenbank-Tabelle. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>FILEGDBIN_READ_OBJECT [TRUE]</code>
Prozedur	FILEGDBIN_CLOSE [][[]]
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>FILEGDBIN_CLOSE</code>

6.6. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von FILEGDBIN.mod
! gelesenen Objekte in der .log Datei an.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG      => DIRECTORY
  MESSAGE     => 'Enter File Geodatabase Input'
  OPT        => input
END_MAP

MAP FILEGDBIN_PARAM
  STATISTICS  => ON
END_MAP

MAP INPUT_SOURCES
  I1 => FILEGDBIN,OPT.input
END_MAP
```

```
MAP INOUT
  I1 => DISPLAY_OBJECT1, IN
END_MAP

| INCL \script\FILEGDBIN.mod
| INCL \script\run1.prg
```

7. Modul GEOJSONIN - GeoJSON lesen

7.1. Allgemeines

Mit dem Modul GEOJSONIN können Objekte aus GeoJSON Dateien gelesen werden. Mit dem Modul können auch JSON Dateien gelesen werden, die keine Erweiterungen für GeoJSON enthalten.

GEOJSONIN wird mit:

```
| INCL \script\geojsonin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

7.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

7.3. Parametermap GEOJSONIN_PARAM

Folgende Parameter können in der Map SHPIN_PARAM für den Modul SHPIN gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF. Objektstatistik am Ende der .log Datei ausgeben.

7.4. Objektmodell

Der Modul SHPIN liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.JSON_FILE(s)	r	Name der Inputdatei ohne Verzeichnis.
IN.JSON_FILENAME(s)	r	Name der Inputdatei ohne Verzeichnis und ohne Endung
IN.JSON_LEVEL(s)	r	Schachtelungstiefe des Objektes. Oberste Objekte haben Schachtelungstiefe 1.
IN.JSON_ELEMENT_PARENT(s)	r	Name des Elternobjektes..
IN.JSON_ELEMENT(s)	r	Name des Objektes.
IN.JSON_GEOM(g)	o	Geometrie des Objektes wenn vorhanden.
IN.<attribute>(*)	o	Attribute des Objektes wenn vorhanden. Können sein vom Typ: Boolean, String, Integer, Real, Map (Structure), List (Arrays)

7.5. Exportierte Prozeduren und Methoden

Prozedur	<code>GEOJSONIN_OPEN ! [s input][[]]</code>
Beschreibung	Öffnet die GeoJSON Datei <input>. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'c:\test\test.json' GEOJSONIN_OPEN</code>
Prozedur	<code>GEOJSONIN_READ_OBJECT ! [][b state]</code>
Beschreibung	Liest das nächste Objekt aus der aktuellen JSON-Inputdatei. Das Objekt wird in der MAP IN zurückgegeben.
Beispiel	<code>GEOJSONIN_READ_OBJECT [TRUE]</code>
Prozedur	<code>GEOJSONIN_CLOSE ! [][[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>GEOJSONIN_CLOSE</code>
Prozedur	<code>JSONIN.FILE_IS_VALID ! [s input][b status]</code>
Beschreibung	Überprüft ob ein File ein gültiges JSON File ist..
Beispiel	<code>'c:\test\test.json' JSONIN.FILE_IS_VALID [TRUE]</code>
Prozedur	<code>JSONIN.SET_EVENT_PROCEDURE ! [&procedure][[]]</code>
Beschreibung	Es kann eine ICS-Procedure gesetzt werden, die nach jedem Event des Lesens des JSON-Files aufgerufen wird. Im IN.Objekt sind Informationen zum Event enthalten.
Beispiel	<pre> PROCEDURE MYJSON_EVENT_PROCEDURE DISPLAY IN.JSON_LEVEL DISPLAY IN.JSON_EVENT DISPLAY IN.JSON_DATA END_PROCEDURE &MYJSON_EVENT_PROCEDURE JSONIN.SET_EVENT_PROCEDURE </pre>
Prozedur	<code>JSONIN.READ_STRING ! [s string][m object]</code>
Beschreibung	Ein JSON String kann in ein ICS-Objekt (MAP) gelesen werden.
Beispiel	<code>'{"de":"Wald","en":"Forest"}' JSONIN.READ_STRING [map]</code>

7.6. Skriptbeispiel

```

! Diese ICS Konfiguration zeigt alle von geojsonin.mod
! gelesenen Objekte in der .log Datei an.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILES
  MESSAGE => 'Enter .json/geojson Input File'
  FILE_FILTER => *

```

```

FILE_EXISTS => TRUE
OPT => input
END_MAP

MAP JSONIN_PARAM
  STATISTICS    => ON
END_MAP

MAP INPUT_SOURCES
  I1 => GEOJSONIN,OPT.input
END_MAP

MAP INOUT
  I1 => GEOJSONIN_DISPLAY_IN0
END_MAP

|INCL \script\GEOJSONIN.mod
|INCL \script\run1.prg

```

8. Modul GMMDBIN - Intergraph GeoMedia ACCESS Datenbank lesen

8.1. Allgemeines

Mit dem Skriptmodul können Objekte aus einer Intergraph GeoMedia Access Datenbank gelesen werden. Der Modul unterstützt speziell das GeoMedia Datenmodell und die GeoMedia Geometrien.

Der Modul wird mit:

```
|INCL \script\gmmdbin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

8.2. Abhängigkeiten von anderen Modulen

Der Modul basiert auf dem Modul DBIN. Alle im Modul DBIN beschriebenen Anteile gelten daher auch für den Modul GMMDBIN. Ziehen Sie deshalb die Dokumentationen des Modules DBIN bei.

8.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.

PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

8.4. Parametermap DBIN_PARAM

Folgende Parameter können in der Map DBIN_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
SELECT_PARSE	o	ON oder OFF, Default = ON. Definiert, ob ein eventuelles SQL-select Statement geparkt werden soll. Als Argument für für das Modul kann ein SQL-select Statement definiert werden. Als Default wird dieses select-Statement von geparkt , um spezielle Attribute zu erkennen, die vom ODBC-Treiber nicht unterstützt werden und speziell gelesen werden müssen. Manchmal können select-Statements (komplexe Joins) nicht erfolgreich geparkt werden. In solchen Fällen ist dieser Parameter auf OFF zu schalten. Sie müssen besorgt sein, dass das select-Statement von ODBC verarbeitet werden kann.

8.5. Parametermap GMMDBIN_PARAM

Zur Zeit stehen keine Parameter in der Map GMMDBIN_PARAM zur Verfügung.

Parameter	req/opt	Beschreibung

8.6. Objektmodell

Der Modul liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.TABLE(s)	r	Tablename des IN Objekts.

Alle weiteren Komponenten sind abhängig von der Tabellen-Definition in der Datenbank.

8.7. Exportierte Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie der Modul DBIN. Ziehen Sie deshalb die Dokumentationen des Modules DBIN bei. Zusätzlich stellt der Modul GMMDBIN folgende Prozeduren und Methoden zur Verfügung.

Prozedur	<code>GMMDBIN_OPEN [s input][]</code>
Beschreibung	Öffnet eine Datenbank definiert und liest Objekte von der Datenbank in Abhängigkeit von <input>. Für <input> können folgende Werte verwendet werden.

*

Liest alle Tabellen der Datenbank.

<tablename>

Liest die Records der definierten Tabelle <tablename>.

<tablename>,<sql-select-statement>

Liest die Records der Tabelle <tablename> entsprechend dem SQL-select-Statement. Beinhaltet das SQL-select-Statement Hochkommas für Strings so sind die Hochkommas ' durch Anführungszeichen " zu ersetzen.

Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
'Buildings,select * from Buildings where Note = "For ski lodge"' GMMDBIN_OPEN
MAP INPUT_SOURCES
  I1 => GMMDBIN,Buildings,'select * from Buildings where Note = "For ski lodge"'
END_MAP
```

Prozedur

GMMDBIN_READ_OBJECT [[b state]

Beschreibung

Liest das nächste IN-Objekt aus der geöffneten Datenbank-Tabelle. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
GMMDBIN_READ_OBJECT [TRUE]
```

Prozedur

GMMDBIN_CLOSE [[]]

Beschreibung

Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
GMMDBIN_CLOSE
```

Der Modul GMMDBIN stellt zusätzlich folgende Prozeduren und Methoden zur Verfügung.

Methode

GEOMEDIA.BLOB_TO_GEOM [B blob-geometry][g geometry]

Beschreibung

Übersetzt eine GeoMedia Access Geometrie in einem Blob in eine ICS-Geometrie.

Beispiel

```
VAR.BLOB GEOMEDIA.BLOB_TO_GEOM [geometry]
```

Methode

GEOMEDIA.BLOB_TO_GEOM [B blob-geometry][(i VAlignment,) (i HAlignment,) (r rotation,) g|li geometry, s type, b status]

Beschreibung

Übersetzt eine GeoMedia Geometrie als BLOB in eine ICS-Geometrie. Kann die Geometry übersetzt werden, wird als Status TRUE zurückgegeben, ansonsten FALSE. Je nach Geometrie-Type wird die Geometry als einzelne Geometry oder als Liste von Geometrien zurückgegeben. Beim GeoMedia-Typ gmpoint wird zusätzlich die Rotation geliefert. Beim GeoMedia-Typ gmtext wird zusätzlich die Rotation, der Text, das horizontale und das vertikale Alignment geliefert. Folgende Typen werden zurückgegeben: point,line,area,list,gmpoint,gmtext.

Beispiel

```
IF IN.Geometry GEOMEDIA.BLOB_TO_GEOM THEN
  => VAR.TYPE
  => VAR.DIM
  => VAR.GEOM
  IF VAR.TYPE = 'gmpoint' THEN
    => VAR.ROT
```

```
ELSIF VAR.TYPE = 'gmtext' THEN
=> VAR.TEXT
=> VAR.ROT
=> VAR.HALI
=> VAR.VALI
END_IF
END_IF
```

Folgende Konversionen werden durchgeführt:

gdbPoint (10)

to point

gdbLinear (1)

to line

gdbAreal (2)

to area

gdbAnySpatial (3)

to point,line,area,gmpoint or gmtext

gdbOrientedPoint (10)

to point and orientation

gdbGraphicsText (5)

to point and orientation, text, horizontal alignment, vertical alignment

collection

list of point,line,area,gmpoint or gmtext

8.8. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von gmmdbin.mod
! gelesenen Objekte in der .log Datei an.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter Access Input Database'
  FILE_FILTER => mdb
  FILE_EXISTS => TRUE
  OPT         => input
END_MAP
```

```
MAP DB_PARAM
  SOURCE      => '' ! ODBC-Source
  USER        => '' ! ODBC-User
  PASSWD      => '' ! ODBC-Password
END_MAP
```

```
MAP DBIN_PARAM
  STATISTICS  => ON
END_MAP
```

```
MAP GMMDBIN_PARAM
```

```

END_MAP

MAP INPUT_SOURCES
  I1 => GMMDBIN, *
END_MAP

MAP INOUT
  I1 => DISPLAY_IN0
END_MAP

|INCL \script\gmmdbin.mod
|INCL \script\db2il\dbdisplay.out
|INCL \script\run1.prg

```

9. Modul GMORAIN - Intergraph GeoMedia Oracle Datenbank lesen

9.1. Allgemeines

Mit dem Skriptmodul können Objekte aus einer Intergraph GeoMedia Oracle Datenbank via ODBC gelesen werden. Der Modul unterstützt speziell das GeoMedia Datenmodell und die GeoMedia Geometrien.

Der Modul wird mit:

```
|INCL \script\gmorain.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

9.2. Abhängigkeiten von anderen Modulen

Der Modul basiert auf den Modulen DBIN und ORAIN. Alle im Modul DBIN und ORAIN beschriebenen Anteile gelten daher auch für den Modul GMORAIN. Ziehen Sie deshalb die Dokumentationen der Module DBIN und ORAIN bei.

9.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

9.4. Parametermap DBIN_PARAM

Folgende Parameter können in der Map DBIN_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
SELECT_PARSE	o	ON oder OFF, Default = ON. Definiert, ob ein eventuelles SQL-select Statement geparkt werden soll. Als Argument für für das Modul kann ein SQL-select Statement definiert werden. Als Default wird dieses select-Statement von geparkt , um spezielle Attribute zu erkennen, die vom ODBC-Treiber nicht unterstützt werden und speziell gelesen werden müssen. Manchmal können select-Statements (komplexe Joins) nicht erfolgreich geparkt werden. In solchen Fällen ist dieser Parameter auf OFF zu schalten. Sie müssen besorgt sein, dass das select-Statement von ODBC verarbeitet werden kann.

9.5. Parametermap ORAIN_PARAM

Bei einer GeoMedia Oracle Datenbank können folgende Parameter in der Map ORAIN_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
SPATIAL_PROCESS	o	ON oder OFF, Default = OFF. Definiert ob Oracle Spatial Geometrien gelesen werden sollen. Mit OFF werden die Geometrien nicht gelesen. Mit ON werden die Geometrien gelesen.

Für die Anwendung der Oracle Spatial Option ist die entsprechende Dokumentation von Oracle zu beachten.

9.6. Parametermap GMORAIN_PARAM

Folgende Parameter können in der Map GMORAIN_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
GDOSYS_OWNER	o	<user>. Definiert bei einer GeoMedia Oracle Datenbank den Benutzer für welchen in den GeoMedia-Metadaten unter GDOSYS die Metadaten verarbeitet werden sollen.

9.7. Objektmodell

Der Modul liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.TABLE(s)	r	Tablename des IN Objekts.
IN.<Geometry>(g)	o	Ein Geometrie-Attribut wird mit demselben Namen wie in der Datenbank geliefert. Der enthaltene Wert entspricht einer ICS-Geometrie vom Typ point, line oder area.

IN.<Geometry>_SDO_GEOMETRY(s)	o	Zusätzlich zu einem Geometrie-Attribut wird in einem Attribut mit dem Suffix _SDO_GEOMETRY die Oracle Spatial Geometrie als String geliefert.
-------------------------------	---	---

Alle weiteren Komponenten sind abhängig von der Tabellen-Definition in der Datenbank.

9.8. Spezielles

Für das Lesen von Oracle Spatial sind folgende Punkte zu beachten.

Oracle Spatial lesen über ODBC mit Oracle PL/SQL

ODBC verarbeitet den Oracle Objekt-Typ MDSYS.SDO_GEOMETRY für die Geometrien nicht. Um die Geometrien über ODBC trotzdem lesen zu können, legt der Modul eine PL/SQL Funktion in Oracle an.

```
create function ILTOOL_SDO_GEOMETRY_TO_STRING (geom MDSYS.SDO_GEOMETRY) RETURN CLOB
```

Diese Funktion wandelt eine Geometrie vom Typ MDSYS.SDO_GEOMETRY in einen String um. Die Funktion wird vom Modul beim Lesen der Geometrien wie folgt angewendet.

```
select ILTOOL_SDO_GEOMETRY_TO_STRING(Geometrie) as Geometrie from Table
```

Die Funktion liefert die Geometrie als String, den der Modul dann in eine ICS-Geometrie umwandelt.

Der Oracle-User aus dem Daten gelesen werden sollen, muss deshalb die Berechtigung haben, eine Funktion anlegen zu können.

9.9. Exportierte Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie die Module DBIN und ORAIN. Ziehen Sie deshalb die Dokumentationen der Module DBIN und ORAIN bei. Zusätzlich stellt der Modul GMORAIN folgende Prozeduren und Methoden zur Verfügung.

Prozedur GMORAIN_OPEN [s input][]

Beschreibung Öffnet eine Datenbank definiert mit DB_PARAM.SOURCE und liest Objekte von der Datenbank in Abhängigkeit von <input>. Für <input> können folgende Werte verwendet werden.

*

Liest alle Tabellen der Datenbank.

<tablename>

Liest die Records der definierten Tabelle <tablename>.

<tablename>,<sql-select-statement>

Liest die Records der Tabelle <tablename> entsprechend dem SQL-select-Statement. Beinhaltet das SQL-select-Statement Hochkommas für Strings so sind die Hochkommas ' durch Anführungszeichen " zu ersetzen.

Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
'Buildings,select * from Buildings where Note = "For ski lodge"' GMORAIN_OPEN
```

```
MAP INPUT_SOURCES
```

```
  I1 => GMORAIN,Buildings,'select * from Buildings where Note = "For ski lodge" '
END_MAP
```

Prozedur	<code>GMORAIN_READ_OBJECT</code> <code>[[b state]</code>
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten Datenbank-Tabelle. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>GMORAIN_READ_OBJECT</code> <code>[TRUE]</code>
Prozedur	<code>GMORAIN_CLOSE</code> <code>[[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>GMORAIN_CLOSE</code>

Der Modul GMORAIN stellt zusätzlich folgende Prozeduren und Methoden zur Verfügung.

Methode	<code>ORACLE.SDO_GEOMETRY_TO_GEOM</code> <code>[s sdo-geometry][(i VAlignment,) (i HAlignment,) (s text,)(r rotation,) g li geometry, i dimension, s type, b status]</code>
Beschreibung	Übersetzt eine Oracle-Spatial Geometrie als String in eine ICS-Geometrie. Kann die Geometry übersetzt werden, wird als Status <code>TRUE</code> zurückgegeben, ansonsten <code>FALSE</code> . Je nach SDO-Type wird die Geometry als einzelne Geometry oder als Liste von Geometrien zurückgegeben. Bei den Typen <code>orientedpoint</code> und <code>GeoMedia-Typ gmpoint</code> wird zusätzlich die Rotation geliefert. Beim <code>GeoMedia-Typ gmtext</code> wird zusätzlich die Rotation, der Text, das horizontale und das vertikale Alignment geliefert. Folgende Typen werden zurückgegeben: <code>point</code> , <code>orientedpoint</code> , <code>line</code> , <code>polygon</code> , <code>multi-point</code> , <code>multiline</code> , <code>multipolygon</code> , <code>collection</code> , <code>gmpoint</code> , <code>gmtext</code> .
Beispiel	<pre> IF IN.Geometrie ORACLE.SDO_GEOMETRY_TO_GEOM THEN => VAR.TYPE => VAR.DIM => VAR.GEOM IF VAR.TYPE = 'orientedpoint' THEN => VAR.ROT ELSIF VAR.TYPE = 'gmpoint' THEN => VAR.ROT ELSIF VAR.TYPE = 'gmtext' THEN => VAR.ROT => VAR.TEXT => VAR.HALI => VAR.VALI END_IF END_IF </pre>

Folgende Konversionen werden durchgeführt:

SDO-point

to point

SDO-orientedpoint

to point and orientation

SDO-line

to line

SDO-polygon

to area

SDO-multipoint

to list of points

SDO-multiline

to list of lines

SDO-multipolygon

to list of areas

SDO-collection

to list of points und/oder lines und/oder areas

GEOMEDIA-SDO-point

to point and orientation

GEOMEDIA-SDO-text

to point and orientation, text, horizontal alignment, vertical alignment

9.10. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von gmorain.mod
! gelesenen Objekte in der .log Datei an.
```

```
|LICENSE \license\iltoolspro.lic
```

```
MAP USER_INPUT1
```

```
    DIALOG      => ODBC
```

```
    OPT         => input
```

```
END_MAP
```

```
MAP DB_PARAM
```

```
    SOURCE      => '' ! ODBC-Source
```

```
    USER        => '' ! ODBC-User
```

```
    PASSWD      => '' ! ODBC-Password
```

```
END_MAP
```

```
MAP DBIN_PARAM
```

```
    STATISTICS  => ON
```

```
END_MAP
```

```
MAP ORAIN_PARAM
```

```
    STATISTICS  => ON
```

```
END_MAP
```

```
MAP GMORAIN_PARAM
```

```
    GDOSYS_OWNER => ''
```

```
END_MAP
```

```
MAP INPUT_SOURCES
```

```
    I1 => GMORAIN,*
```

```
END_MAP
```

```
MAP INOUT
```

```
    I1 => DISPLAY_IN0
```

```
END_MAP
```

```
|INCL \script\gmorain.mod
```

```
|INCL \script\db2il\dbdisplay.out
```

```
|INCL \script\run1.prg
```

10. Modul IFCIN - Industry Foundation Classes IFC lesen

10.1. Allgemeines

Mit dem Modul können Objekte aus IFC-Files - STEP Physical File (SPF) *.ifc - gelesen werden.

Das Modul basiert auf dem IFC SDK der Open Design Alliance (ODA).

Der Modul wird mit:

```
| INCL \script\odaifcin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

10.2. Parametermap IFCIN_PARAM

Folgende Parameter können in der Map IFCIN_PARAM Parameter für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF. Default OFF. Statistik anzeigen.

10.3. Objektmodell

Allgemeine Komponenten in jedem IN-Objekt

Komponente	req/opt	Beschreibung
IN. IFC_ID(i)	r	IFC ID des Objektes.
IN. IFC_CLASS(s)	r	IFC Class Objektes.
IN. IFC_NAME(s)	o	Name des Objektes
IN. IFC_GEOM(g)	o	Geometrie des Objektes falls vorhanden.
IN. IFC_PROPERTY-SETS(li)	o	Propertysets des Objektes falls vorhanden.
IN. * (*)	o	Weitere Attribute zur IFC-Klasse, wie diese in der Beschreibung der IFC Klassen aufgeführt sind. Siehe zum Beispiel https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/

Listenelemente für OUT. IFC_PROPERTYSETS

Beinhaltet eine Liste vom Maps mit den Propertysets.

Jede Map eines Propertysets beinhaltet einen Namen und eine Liste mit den einzelnen Propertyinglevalues.

Komponente	req/opt	Beschreibung
<m>. IFC_ID(i)	r	IFC ID des Propertysets.
<m>. IFC_CLASS(s)	r	IFC Class des Propertysets. Immer IFCPROPERTYSET .
<m>. IFC_NAME(s)	r	Name des Propertysets.

<m>.IFC_PROPERTYSET_SINGLEVALUES(<i>li</i>)	r	Liste mit Maps mit den Property Singlevalues des Propertysets. Jeder Property Singlevalue mit Name und Wert in der Map.
<m>.*(*)	o	Weitere Attribute zur IFC-Klasse, wie diese in der Beschreibung der IFC Klassen aufgeführt sind. Siehe zum Beispiel https://standards.buildingsmart.org/IFC/RELEA-SE/IFC4/ADD2_TC1/HTML/

Listenelemente für OUT.IFC_PROPERTYSETS.IFC_PROPERTYSET_SINGLEVALUES

Beinhaltet eine Liste vom Maps mit den Property Singlevalues.

Jede Map eines Property Singlevalues beinhaltet einen Namen mit Wert.

Komponente	req/opt	Beschreibung
<m>.IFC_ID(<i>i</i>)	r	IFC ID des Property Singlevalue.
<m>.IFC_CLASS(<i>s</i>)	r	IFC Class des Property Singlevalue. Immer IFCPROPERTYSET_SINGLEVALUE .
<m>.IFC_NAME(<i>s</i>)	r	Name des Property Singlevalue.
<m>.IFC_SINGLEVALUE(<i>*</i>)	r	Wert des Property Singlevalue.
<m>.*(*)	o	Weitere Attribute zur IFC-Klasse, wie diese in der Beschreibung der IFC Klassen aufgeführt sind. Siehe zum Beispiel https://standards.buildingsmart.org/IFC/RELEA-SE/IFC4/ADD2_TC1/HTML/

10.4. Exportierte Prozeduren und Methoden

Prozedur	IFCIN_OPEN ! [<i>s input</i>][<i>i</i>]
Beschreibung	Öffnet das IFC-File <input> und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'c:\temp\test.ifc' IFCIN_OPEN</code>
Prozedur	IFCIN_READ_OBJECT ! [<i>i</i>][<i>b state</i>]
Beschreibung	Liest das nächste IN-Objekt aus dem aktuellen Inputfile. Falls kein Objekt mehr gelesen werden kann, wird der Status FALSE geliefert. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>IFCIN_READ_OBJECT [TRUE]</code>
Prozedur	IFCIN_CLOSE ! [<i>i</i>][<i>i</i>]
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>IFCIN_CLOSE</code>
Prozedur	IFCIN_DISPLAY_IN0 ! [<i>i</i>][<i>i</i>]
Beschreibung	Displays das IN-Object mit allen Komponenten und Subkomponenten.
Beispiel	<code>IFCIN_DISPLAY_IN0</code>

10.5. Skriptbeispiel

Beispiel ohne Datenbankanbindung.

```
! Diese ICS Konfiguration zeigt alle von ifcin.mod
! gelesenen Objekte in der .log Datei an.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE
  MESSAGE => 'Enter .ifc Input File'
  FILE_FILTER => ifc
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP IFCIN_PARAM
  STATISTICS      => ON
END_MAP

MAP INPUT_SOURCES
  I1 => IFCIN,OPT.input
END_MAP

MAP INOUT
  I1              => IN.IFC_CLASS
  I1,IFCSPACE    => IFCIN_DISPLAY_IN0
  I1,*           => IFCIN_DISPLAY_IN0
END_MAP

MAP MACRO
END_MAP

|INCL \script\odaifcin.mod
|INCL \script\run1.prg
```

11. Modul IL2IN - INTERLIS 2 lesen

11.1. Allgemeines

Mit dem Modul IL2IN können Objekte aus einer INTERLIS 2 .xtf Datei gelesen werden. Der Modul interpretiert neben der .xtf Datei auch die zugehörigen INTERLIS 2 Datenmodelle (.ili Dateien). Jedes Objekt wird von IL2IN auf seine Konsistenz gegenüber den INTERLIS 2 Datenmodellen überprüft. Falls z.B. zwingende Attribute vergessen oder falsche Attributwerte gefunden werden, werden entsprechende Fehlermeldungen ausgegeben.

IL2IN wird mit:

```
|INCL \script\il2in.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

11.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

11.3. Parametermap IL2IN_PARAM

Folgende Parameter können in der Map IL2IN_PARAM für den Modul IL2IN gesetzt werden:

Parameter	req/opt	Beschreibung
FORCE_COMPILE	r	Falls dieser Parameter auf ON (Default = OFF) gesetzt wird, werden die INTERLIS Datenmodelle jedes mal mit dem INTERLIS 2.2 Compiler analysiert. Falls die Option auf OFF gesetzt ist, wird zuerst nachgeschaut ob bereits eine vorcompilierte Version des Modells existiert (.ilo und .ilp Datei). Das Laden von vorcompilierten Modelldateien ist viel schneller möglich, daher ist die Option normalerweise auf OFF gesetzt.
MODELS	r	<p>Mit dieser Option kann man angeben, wie die zum Datensatz gehörigen Modelldateien bestimmt werden. XTF bedeutet, dass die Modelle automatisch aus der gewählten .xtf Datei bestimmt werden. Dazu wird wie folgt vorgegangen:</p> <ul style="list-style-type: none"> • Es wird nach dem letzten ALIAS ENTRIES Element der HEADER-SECTION gesucht und aus diesem der Name des Hauptmodells gelesen. Falls keine ALIAS ENTRIES in der .xtf Datei vorhanden sind, wird der Modellname des Hauptmodells aus dem ersten Objekt-Tag der DATASECTION extrahiert. • Das Hauptmodell wird zuerst in \user\models2 dann in \system\models2 gesucht. Falls das Hauptmodell gefunden wird, wird das Hauptmodell nach IMPORTS durchsucht und so die allfälligen Basismodelle bestimmt. Für die Basismodelle wird ebenso verfahren. • Der INTERLIS 2.2 Compiler wird automatisch mit den so bestimmten Modellen aufgerufen. <p>Normalerweise funktioniert das oben beschriebene Verfahren gut und ist ausserdem sehr praktisch in der Benutzung. Es kann aber sein, dass man in bestimmten Fällen die automatische Bestimmung der Datenmodelle ausschalten möchte. In diesem Fall kann man unter MODELS auch eine Liste von Datenmodelldateien in der korrekten Reihenfolge für den INTERLIS 2.2 Compiler angeben. Beispiel:</p> <pre>MODELS => units.ili,time.ili,coordsys.ili,dm01avch24d.ili</pre> <p>Die Dateien werden ebenfalls zuerst in \user\models2 dann in \system\models2 gesucht.</p>
MODEL_NAME	r	Unter MODEL_NAME kann man den Namen eines Basismodells des Hauptmodells angeben (z.B. DM01AVCH24D für eine kantonale Erweiterung). Das Lesen der Daten findet dann nur gemäss den Regeln des Basismodells statt (polymorpher Leser). Hinweis: Dieses Feature ist noch nicht verfügbar, weil die benutzte C-Bibliothek IOM (INTERLIS Object Model) dieses Feature noch nicht vollständig unterstützt.

MODEL_DIR	r	Normalerweise werden Modelldateien (.ili) zuerst in \user\models2 ge-sucht. Mit der Option MODEL_DIR kann man einen alternativen (User-)Suchpfad angeben (Default = OFF).
DECODER	o	UTF8 Texte decodieren (ON / OFF, Default ON).
TRACE	r	Falls dieser Parameter auf ON (Default = OFF) gesetzt wird, wird für jedes von IOM gelesene Objekt (IOM = INTERLIS Object Model) eine Meldung in die Logdatei ausgegeben.
STATISTICS	o	Objektstatistik am Ende der .log Datei ausgeben (Default = OFF).
DEBUG	o	ON oder OFF. Debugmodus ein oder aus (Default = OFF).

11.4. Objektmodell

Komponenten für jedes IN-Objekt

Komponente	req/opt	Beschreibung
IN.MODEL(s)	r	Model des IN Objekts.
IN.TOPIC(s)	r	Topic des IN-Objekt
IN.CLASS(s)	r	Tabelle des IN-Objekts.
IN.TID(s)	r	Transferidentifikation des IN-Objekts .
IN.LINE(i)	r	Zeilennummer des Objekts in der Inputdatei.

Weitere Komponenten

Die restlichen Objektkomponenten sind abhängig vom dazugehörigen INTERLIS Modell (s.a. IN.MODEL, IN.TOPIC bzw. IN.CLASS). Alle INTERLIS Attribute werden als Komponenten des IN-Objekts mit dem gleichem Namen geliefert. Die INTERLIS Datentypen werden wie folgt auf ICS Datentypen abgebildet:

INTERLIS Datentyp	ICS Datentyp
NUMBER	real oder int.
TEXT	string.
ENUMERATION	string.
STRUCTURE	map. Das XML-Tag der Struktur kann dem Label der Map entnommen werden (mit GET_LABEL).
LIST	list of map. Die XML-Tags der Strukturelemente können den Labeln der Maps entnommen werden.
BAG	list of map. Die XML-Tags der Strukturelemente können den Labeln der Maps entnommen werden.
ROLE	link.

11.5. Exportierte Prozeduren und Methoden

Prozedur IL2IN_OPEN ! [s input][]

Beschreibung Öffnet eine bestehende INTERLIS 2 Datei <input> und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel IL2IN_OPEN

Prozedur	<code>IL2IN_READ_OBJECT ! [b state][[]]</code>
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten INTERLIS 2 Datei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>IL2IN_READ_OBJECT [TRUE]</code>
Prozedur	<code>IL2IN_RESET_READ_REF ! [s tid][[]]</code>
Beschreibung	Bereitet das Lesen von Objekten, welche mit dem Objekt mit der Transferidentifikation <tid> via ASSOCIATION's verbunden sind, vor.
Beispiel	<code>IN.TID IL2IN_RESET_READ_REF []</code>
Prozedur	<code>IL2IN_READ_NEXT_REF ! [s tid][o object,b state]</code>
Beschreibung	Liest das nächste Objekt, welches via eine ASSOCIATION mit dem Objekt mit der Transferidentifikation <tid> verbunden ist.
Beispiel	<pre>! display all objects linked with IN.TID IN.TID IL2IN_RESET_READ_REF WHILE IN.TID IL2IN_READ_NEXT_REF DO => VAR.OBJECT DISPLAY VAR.OBJECT END_WHILE</pre>
Prozedur	<code>IL2IN_CLOSE ! [][[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>IL2IN_CLOSE</code>

11.6. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von il2in.mod
! gelesenen Objekte in der .log Datei an.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .xtf Input File'
  FILE_FILTER => xtf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP
```

```
MAP IL2IN_PARAM
  INPUT_EXTENSION => xtf
  MODELS           => XTF
  MODEL_NAME      => OFF
  FORCE_COMPILE    => OFF
  TRACE           => OFF
  STATISTICS      => ON
END_MAP
```

```
MAP INPUT_SOURCES
  I1 => IL2IN,OPT.input
```

```

END_MAP

MAP INOUT
  I1 => DISPLAY_OBJECT1, IN
END_MAP

| INCL \script\util.lib
| INCL \script\il2in.mod
| INCL \script\run1.prg

```

12. Modul ILIN - INTERLIS 1 lesen

12.1. Allgemeines

Mit dem Modul ILIN können Objekte aus einer INTERLIS 1 .itf Datei gelesen werden. Der Modul interpretiert neben der .itf Datei auch die zugehörigen INTERLIS 1 Datenmodelle (.ili Dateien). Jedes Objekt wird vom Modul auf seine Konsistenz gegenüber den INTERLIS 1 Datenmodellen überprüft. Falls z.B. zwingende Attribute vergessen oder falsche Attributwerte gefunden werden, werden entsprechende Fehlermeldungen ausgegeben.

Der Modul wird mit:

```
| INCL \script\ilin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

12.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

12.3. Parametermap ILIN_PARAM

Folgende Parameter können in der Map ILIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
INTERLIS_DEF	r	Dateiname der INTERLIS Modelldatei. Die Angabe des Parameters ist obligatorisch
METHA_OBJECTS	r	ON oder OFF, Default = OFF. Methaobjekte liefern ein oder aus.
LOG_TABLE	r	ON oder OFF, Default = ON. Für jede gelesene Tabelle eine Meldung ausgeben.
TRACE	r	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.
ENUM_TO_TEXT	r	ON oder OFF, Default = OFF. Aufzählungsattribute auch als Textversion liefern. Die Textversion des Attributs ist in IN.<Attribut>_TXT verfügbar.
CALC_SURFACE	r	ON oder OFF, Default = OFF. SURFACE-Geometrien berechnen und dem Hauptobjekt zuordnen.
VALUE_CHECK	r	ON oder OFF, Default = ON. Wertebereichstests auf Attributen durchführen.
CHARSET_CHECK	r	ON oder OFF, Default = OFF. Zeichensatz gemäss Norm SN612030 überprüfen.

MATH_DEGREES	r	ON oder OFF, Default = OFF. DEGREES im mathematischen Sinn interpretieren, d.h. 0.0 = horizontal, Orientierung = Gegenuhrzeigersinn.
ARC_CHECK	r	ON oder OFF, Default = OFF. Kreisbogengeometrie testen.
DOUBLEPOINT_CHECK	r	ON oder OFF, Default = OFF. Nacheinanderfolgende doppelte Punkte in Linien testen.
TOPICS	r	topic[, topic]. Nur Topics gemäss Liste in TOPICS lesen. Die Topics müssen als kommaseparierete Liste (z.B. Fixpunkte, Bodenbedeckung) angegeben werden.
SYNTAX_ERROR_HALT	r	ON oder OFF, Default = OFF. Bei einem Syntax-Error in der INTERLIS-Datei wird das Weiterlesen abgebrochen.
DEBUG	r	ON oder OFF, Default = OFF. Debugmodus ein oder aus.
SAVE_REF	r	ON oder OFF, Default = ON. INTERLIS Referenzen unterstützen. Falls die Option eingeschaltet ist können INTERLIS Referenzen (->) in der Skriptsprache durch Angabe des vollständigen Pfad (z.B. IN.Objekt.Nummer) aufgelöst werden. Dazu müssen die referenzierten Objekte zwischengespeichert werden. Falls die Option auf OFF gesetzt ist, werden die referenzierten Objekte nicht zwischengespeichert. Die Auflösung der Referenzen via den Pfad ist dann nicht mehr möglich, dafür ist das Lesen der Inputdatei schneller.
STATUS	r	Enthält nach dem Lesen eines Objektes einen Fehlerstatus, falls ein Fehler aufgetreten ist.
TAKE_RANGE	o	topic[, topic]. Nur Range berechnen für Topics, die in der Liste TAKE_RANGE sind. Die zu berücksichtigen Topics müssen als kommaseparierete Liste (z.B. Fixpunkte, Bodenbedeckung) angegeben werden. ILIN berechnet für die gelesenen Objekte die maximale Ausdehnung und stellt diese in ILIN_PARAM mit MIN_X, MIN_Y, MAX_X, MAX_Y zur Verfügung. Diese Ausdehnung kann/wird von Konfigurationen weiterverarbeitet. Mit diesem Parameter kann die Berechnung des Ranges auf spezifizierte Topics eingeschränkt werden.
IGNORE_RANGE	o	topic[, topic]. Nur Range berechnen für Topics, die nicht in der Liste IGNORE_RANGE sind. Gegenteil von TAKE_RANGE. Ist TAKE_RANGE definiert, wird dieser Parameter ignoriert.
STATISTICS	r	ON oder OFF, Default = OFF. Statistik anzeigen.
STATISTICS_FILE	o	ON oder OFF, Default = OFF. Statistik mit File.
STATISTICS_MODEL	o	ON oder OFF, Default = OFF. Statistik mit Model.

12.4. Objektmodell

Allgemeine Komponenten für jedes IN-Objekt

Komponente	req/opt	Beschreibung
IN.MODEL(s)	r	Model des IN Objekts.
IN.TOPIC(s)	r	Topic des IN-Objekt
IN.TABLE(s)	r	Table des IN-Objekts.
IN.OBJID(s)	r	Transferidentifikation des IN-Objekts .
IN.LINE(i)	r	Zeilennummer des Objekts in der Inputdatei.

Normalerweise wird vom Modul pro OBJE-Zeile ein Objekt geliefert. Falls der Parameter ILIN_PARAM.METHA_OBJECTS auf 'ON' gesetzt wurde, werden auch für die INTERLIS Label MODL, EMOD, TOPI, ETOP, TABL, ETAB Objekte zurückgeliefert (sog. Methaobjekte).

Zusätzliche Objekte und Komponenten für Typ SURFACE Geometrietabellen

Für INTERLIS-Tabellen die implizit aus SURFACE-Deklarationen entstehen (z.B. Fixpunkte.LFPNachfuehrung_Perimeter) sind folgende zusätzlichen Objekte und Komponenten verfügbar:

Objekt	req/opt	Beschreibung
<Maintable>_ <Geometry-Attribute>	r	Implizite INTERLIS Table gebildet aus der Haupttabelle mit dem SURFACE-Attribute und dem Namen des SURFACE Attributes.

Komponente	req/opt	Beschreibung
IN.GEOM(1)	r	Geometrie des Objekts.
IN.REFID(s)	r	Referenz auf das Hauptobjekt.

Falls der Parameter ILIN_PARAM.CALC_SURFACE auf ON gesetzt wurde, werden alle SURFACE Flächen direkt als Attribut zum Objekt geliefert. Falls ausserdem eine LINEATTR Definition für die SURFACE im Datenmodell definiert wurde, wird das LINATTR Attribut als GATTR (s.a. ICS.GET_GATTR) zu den Randlinien der Fläche gespeichert. Bei mehreren LINEATTR Attributen pro SURFACE wird das erste Attribut vom Grundtyp INTEGER als GATTR geliefert.

Zusätzliche Objekte und Komponenten für Typ AREA Geometrietabellen

Für INTELIS-Tabellen die implizit aus AREA-Deklarationen entstehen (z.B. Bodenbedeckung.BoFlaeche_Geometrie) sind folgende zusätzlichen Objekte und Komponenten verfügbar:

Objekt	req/opt	Beschreibung
<Maintable>_ <Geometry-Attribute>	r	Implizite INTERLIS Table gebildet aus der Haupttabelle mit dem AREA-Attribute und dem Namen des AREA Attributes.

Komponente	req/opt	Beschreibung
IN.GEOM(1)	r	Geometrie des Objekts.

Die restlichen Objektkomponenten sind abhängig von der dazugehörigen INTERLIS Table (s.a. IN.MODEL, IN.TOPIC bzw. IN.TABLE). Alle INTERLIS Attribute werden als Komponenten des IN-Objekts mit dem gleichem Namen geliefert. Die INTERLIS Datentypen werden wie folgt auf ICS Datentypen abgebildet:

INTERLIS Datentyp	ICS Datentyp
IRANGE	int.
RRANGE	real.
Text	string.
GRADS	real.
DEGREES	real.
RADIANS	real.
ENUMERATION	int. string bei ILIN_PARAM.ENUM_TO_TEXT=ON

COORD2	point.
COORD3	point.
POLYLINE	line.
SURFACE	area. bei ILIN_PARAM.CALC_SURFACE=ON
AREA	point. Zentroid.
-> (Referenz)	ilink.

Referenzen von INTERLIS-Objekten auf andere INTERLIS-Objekte (z.B. Hoehen.Entstehung : -> HoehenNachfuehrung) sind als spezieller Datentyp `ilink` implementiert. INTERLIS-Referenzen können aufgelöst werden, indem deren Komponenten in iG/Script angesprochen werden (z.B. `IN.Entstehung.Identifikator`).

Falls mit der Parameter `ILIN_PARAM.CALC_SURFACE` auf `ON` gesetzt wurde, werden alle SURFACE Flächen direkt als Attribut zum Objekt geliefert.

Falls ausserdem eine `LINEATTR` Definition für die SURFACE im Datenmodell definiert wurde, wird das `LINATTR` Attribut als `GATTR` (s.a. `ICS.GET_GATTR`) zu den Randlinien der Fläche gespeichert. Bei mehreren `LINEATTR` Attributen pro SURFACE wird das erste Attribut vom Grundtyp `INTEGER` als `GATTR` geliefert.

12.5. Exportierte Prozeduren und Methoden

Prozedur	<code>ILIN_OPEN [s input][[]]</code>
Beschreibung	Öffnet eine bestehende INTERLIS 1 Datei <input> und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'test.itf' ILIN_OPEN</code>
Prozedur	<code>ILIN_READ_OBJECT [][b state]</code>
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten INTERLIS 1 Datei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>ILIN_READ_OBJECT [TRUE]</code>
Prozedur	<code>ILIN_CLOSE [][[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>ILIN_CLOSE</code>
Methode	<code>ILIN.GET_ILINK_KEY [ilink il] [s key]</code>
Beschreibung	Schlüssel aus Beziehungsattribut lesen.
Beispiel	<code>IN.Entstehung ILIN.GET_ILINK_KEY ['200015']</code>
Methode	<code>ILIN.GET_ILINK_TABLE [ilink il] [s table]</code>
Beschreibung	Tabellenname der referenzierten Tabelle aus Beziehungsattribut lesen.
Beispiel	<code>IN.Objekt ILIN.GET_ILINK_TABLE ['LFPNachfuehrung']</code>
Methode	<code>ILIN.GET_ILINK_OBJECT [ilink il] [m object]</code>
Beschreibung	Liefert zu einem <code>ilink</code> das referenzierte Objekt als Map.

Beispiel	<code>IN.Objekt ILIN.GET_ILINK_OBJECT [map]</code>
Methode	<code>ILIN.GET_OBJECT_ILINK_LIST [m object, i linkdepthmax] [li list]</code>
Beschreibung	Liefert zu einem Input-Objekt - enthalten in der Map object - alle Objekte als Maps in einer Liste list zurück, auf die das Input-Objekt über Beziehungsattribute referenziert. Falls das Input-Objekt keine Beziehungsattribute aufweist, ist die Liste leer. Mit den Argument linkdepthmax kann die Tiefe der zu berücksichtigen Beziehungen definiert werden. Die Tiefe 0 liefert keine Objekte zurück. Die Tiefe 1 liefert alle Objekte zurück, die direkt vom Input-Objekt referenziert werden. Die Tiefe 2 liefert alle Objekte zurück, die direkt vom Input-Objekt referenziert werden und diejenigen, die von diesen vom Input-Objekte referenzierten Objekte wiederum referenzieren. Und so weiter.
Beispiel	<code>&IN 5 ILIN.GET_OBJECT_ILINK_LIST [list]</code>
Methode	<code>ILIN.COMPILE [s modelldatei] [s modell,b status]</code>
Beschreibung	Modelldatei <modelldatei> (.ili) mit INTERLIS-Compiler compilieren. Falls das Modell keine Fehler enthält, wird TRUE und der Name des Modells auf dem Stack geliefert, sonst FALSE. ILIN.COMPILE erzeugt ein Abbild des INTERLIS-Datenmodells in der vordefinierten Map ILIN_MODEL.
Beispiel	<code>'td.ili' ILIN.COMPILE ['Grunddatensatz',TRUE]</code>
Methode	<code>ILIN.GET_MODEL [s modelldatei] [s modell,b status]</code>
Beschreibung	Modellnamen aus der .ili Datei <modelldatei> lesen. Falls der Modellname gelesen werden konnte, wird TRUE und der Name des Modells auf dem Stack geliefert, sonst FALSE. ILIN.GET_MODEL ist für die Bestimmung des Modellnamens die effizientere Variante als ILIN.COMPILE. ILIN.GET_MODEL füllt jedoch im Gegensatz zu ILIN.COMPILE die Map ILIN_MODEL nicht.
Beispiel	<code>'td.ili' ILIN.GET_MODEL ['Grunddatensatz',TRUE]</code>
Methode	<code>ILIN.SET_ALTRANGE [r minx,r miny,r minz,r maxx,r maxy,r maxz] []</code>
Beschreibung	Alternativen Koordinatenbereich für Korrdinatenbereichtests festlegen. Der neue Koordinatenbereich übersteuert die aus dem INTERLIS-Datenmodell gelesenen Koordinatenbereiche.
Beispiel	<code>600000.0 200000.0 500.0 650000.0 250000.0 600.0 ILIN.SET_ALTRANGE []</code>

12.6. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von ilin.mod
! gelesenen Objekte in der .log Datei an.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
```

```

END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF    => \models\Grunddatensatz.ili
  TRACE          => OFF
  STATISTICS     => ON
END_MAP

MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
END_MAP

MAP INOUT
  I1 => DISPLAY_OBJECT1,IN
END_MAP

|INCL \script\ilin.mod
|INCL \script\run1.prg

```

13. Modul ILTOPO - INTERLIS 1 lesen mit Topologieberechnung

13.1. Allgemeines

Mit dem Modul ILTOPO können ICS Objekte aus einer INTERLIS 1 .itf Datei gelesen werden. Der Modul interpretiert neben der .itf Datei auch das zugehörige INTERLIS 1 Datenmodell (.ili Dateien). Jedes Objekt wird vom Modul auf seine Konsistenz gegenüber dem INTERLIS 1 Datenmodell überprüft. Falls z.B. zwingende Attribute vergessen oder falsche Attributwerte gefunden werden, werden entsprechende Fehlermeldungen ausgegeben. Zudem können für den INTERLIS-Type AREA die Flächen (geschlossene Polygone) berechnet werden. Der Modul entspricht weitgehend dem Modul ILIN mit zusätzlich eingebauter automatischer Topologieberechnung.

Der Modul wird mit:

```
|INCL \script\iltopo.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

13.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

13.3. Parametermap ILIN_PARAM

Folgende Parameter können in der Map ILIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
INTERLIS_DEF	r	Filename des INTERLIS Definitionsfiles. Die Angabe des Parameters ist obligatorisch
METHA_OBJECTS	r	ON oder OFF, Default = OFF. Methaobjekte liefern ein oder aus.

LOG_TABLE	r	ON oder OFF, Default = ON. Für jede gelesene Tabelle eine Meldung ausgeben.
TRACE	r	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.
ENUM_TO_TEXT	r	ON oder OFF, Default = OFF. Aufzählungsattribute auch als Textversion liefern. Die Textversion des Attributs ist in IN.<Attribut>_TXT verfügbar.
CALC_SURFACE	r	ON oder OFF, Default = OFF. SURFACE-Geometrien berechnen und dem Hauptobjekt zuordnen.
VALUE_CHECK	r	ON oder OFF, Default = ON. Wertebereichtests auf Attributen durchführen.
CHARSET_CHECK	r	ON oder OFF, Default = OFF. Zeichensatz gemäss Norm SN612030 überprüfen.
MATH_DEGREES	r	ON oder OFF, Default = OFF. DEGREES im mathematischen Sinn interpretieren, d.h. 0.0 = horizontal, Orientierung = Gegenuhrzeigersinn.
ARC_CHECK	r	ON oder OFF, Default = OFF. Kreisbogengeometrie testen.
DOUBLEPOINT_CHECK	r	ON oder OFF, Default = OFF. Nacheinanderfolgende doppelte Punkte in Linien testen.
TOPICS	r	Nur Topics gemäss Liste in TOPICS lesen. Die Topics müssen als kommaseparierte Liste (z.B. Fixpunkte, Bodenbedeckung) angegeben werden.
SYNTAX_ERROR_HALT	r	ON oder OFF, Default = OFF. Bei einem Syntax-Error in der INTERLIS-Datei wird das Weiterlesen abgebrochen.
DEBUG	r	ON oder OFF, Default = OFF. Debugmodus ein oder aus.
SAVE_REF	r	ON oder OFF, Default = ON. INTERLIS Referenzen unterstützen. Falls die Option eingeschaltet ist, können INTERLIS Referenzen (->) in der Skriptsprache durch Angabe des vollständigen Pfad (z.B. IN.Objekt.Nummer) aufgelöst werden. Dazu müssen die referenzierten Objekte zwischengespeichert werden. Falls die Option auf OFF gesetzt ist, werden die referenzierten Objekte nicht zwischengespeichert. Die Auflösung der Referenzen via den Pfad ist dann nicht mehr möglich, dafür ist das Lesen der Inputdatei schneller.
STATUS	r	Enthält nach dem Lesen eines Objekts einen Fehlerstatus, falls ein Fehler aufgetreten ist.
STATISTICS	r	ON oder OFF, Default = OFF. Statistik anzeigen.
STATISTICS_FILE	o	ON oder OFF, Default = OFF. Statistik mit File.
STATISTICS_MODEL	o	ON oder OFF, Default = OFF. Statistik mit Model.

13.4. Parametermap ILIN_TOPO

In der Map ILIN_TOPO sind die gewünschten Geometrie-Tabellen des Types AREA einzutragen, für welche die Topologie berechnet werden sollen.

Parameter	req/opt	Beschreibung
Topic, Table	r	AREA, BOUNDARY, BAD. Mit dem Eintrag der Table der Begrenzungslinien eines AREA Typs wird definiert, dass für dieses Objekt die Topologie berechnet werden soll. Die Topologieberechnung berech-

	<p>net aus den Begrenzungslinien und den Zentroiden die Flächen. Die Einträge haben folgende Bedeutung:</p> <p>AREA</p> <p>Mit dem Code AREA wird definiert, dass die berechneten Flächen als Objekte zurückgegeben werden sollen.</p> <p>BOUNDARY</p> <p>Mit dem Code BOUNDARY wird definiert, dass die berechneten, gültigen Begrenzungslinien als Objekte zurückgegeben werden sollen.</p> <p>BAD</p> <p>Mit dem Code BAD wird definiert, dass die berechneten ungültigen Begrenzungslinien als Objekte zurückgegeben werden sollen.</p> <p>AREA, BOUNDARY, BAD können einzeln oder zusammen kommasepariert aufgeführt sein.</p>
--	--

Beispiel A.1. ILIN_TOPO Definitionen

```
MAP ILIN_TOPO
  Bodenbedeckung,BoFlaeche_Geometrie => AREA,BOUNDARY
  Liegenschaften,Liegenschaft_Geometrie => AREA,BOUNDARY
END_MAP
```

Die obigen Definitionen berechnen für die Tables BoFlaeche_Geometrie und Liegenschaft_Geometrie die Topologie des Types AREA aufgrund der Zentroide und der Begrenzungslinien. Der Modul liefert als zusätzliche Objekte die berechneten Objekte in den Tabellen BoFlaeche_Area, BoFlaeche_Boundary, Liegenschaft_Area und Liegenschaft_Boundary.

13.5. Objektmodell

Allgemeine Komponenten für jedes IN-Objekt

Komponente	req/opt	Beschreibung
IN.MODEL(s)	r	Model des IN Objekts.
IN.TOPIC(s)	r	Topic des IN-Objekt
IN.TABLE(s)	r	Table des IN-Objekts.
IN.OBJID(s)	r	Transferidentifikation des IN-Objekts .
IN.LINE(i)	r	Zeilennummer des Objekts in der Inputdatei.

Normalerweise wird vom Modul pro OBJE-Zeile ein Objekt geliefert. Falls der Parameter ILIN_PARAM.METHA_OBJECTS auf ON gesetzt wurde, werden auch für die INTERLIS Label MODL, EMOD, TOPI, ETOP, TABL, ETAB Objekte zurückgeliefert (sog. Methaobjekte).

Zusätzliche Objekte und Komponenten für Typ SURFACE Geometrietabellen

Für INTERLIS-Tabellen die implizit aus SURFACE-Deklarationen entstehen (z.B. Fixpunkte.LFPNachfuehrung_Perimeter) sind folgende zusätzlichen Objekte und Komponenten verfügbar:

Objekt	req/opt	Beschreibung
--------	---------	--------------

<Maintable>_ <Geometry-Attribute>	r	Implizite INTERLIS Table gebildet aus der Haupttabelle mit dem SURFACE-Attribute und dem Namen des SURFACE Attributes.
--------------------------------------	---	--

Komponente	req/opt	Beschreibung
IN.GEOM(1)	r	Geometrie des Objekts.
IN.REFID(s)	r	Referenz auf das Hauptobjekt.

Falls der Parameter `ILIN_PARAM.CALC_SURFACE` auf `ON` gesetzt wurde, werden alle `SURFACE` Flächen direkt als Attribut zum Objekt geliefert. Falls ausserdem eine `LINEATTR` Definition für die `SURFACE` im Datenmodell definiert wurde, wird das `LINATTR` Attribut als `GATTR` (s.a. `ICS.GET_GATTR`) zu den Randlinien der Fläche gespeichert. Bei mehreren `LINEATTR` Attributen pro `SURFACE` wird das erste Attribut vom Grundtyp `INTEGER` als `GATTR` geliefert.

Zusätzliche Objekte und Komponenten für Typ AREA Geometrietabellen

Für INTERLIS-Tabellen die implizit aus AREA-Deklarationen entstehen (z.B. Bodenbedeckung.BoFlaeche_Geometrie) sind folgende zusätzlichen Objekte und Komponenten verfügbar:

Objekt	req/opt	Beschreibung
<Maintable>_ <Geometry-Attribute>	r	Implizite INTERLIS Table gebildet aus der Haupttabelle mit dem AREA-Attribute und dem Namen des AREA Attributes.

Komponente	req/opt	Beschreibung
IN.GEOM(1)	r	Geometrie des Objekts.

Zusätzliche Objekte und Komponenten für ILIN_TOPO

Falls in der Map `ILIN_TOPO` die entsprechenden Einträge für den INTERLIS Typ AREA vorhanden sind, werden von der Topologieberechnung folgende zusätzlichen Objekte zur Verfügung gestellt:

AREA

Berechnete Flächen werden zur Verfügung gestellt.

```
MAP ILIN_TOPO
  Bodenbedeckung,BoFlaeche_Geometrie => AREA
END_MAP
```

Objekt	Beschreibung
<Table>_Area	Zusätzliches Objekt wie das Objekt mit dem Zentroid mit der Extension <code>_Area</code> . Das Objekt beinhaltet alle Komponenten wie das Objekt mit dem Zentroid plus einer Komponente <code>IN.GEOM</code> mit der berechneten Fläche. Beispiel: <code>BoFlaeche</code> ist das Objekt mit dem Zentroid. <code>BoFlaeche_Area</code> ist das Objekt mit der zusätzlichen Komponente der berechneten Fläche.

Komponente	Beschreibung
IN.GEOM(a)	Berechnete Fläche
*	Alle weiteren Komponenten sind identische mit den Komponenten des Objekts mit dem Zentroid.

BOUNDARY

Berechnete, gültige Bergrenzungslinien werden zur Verfügung gestellt.

```
MAP ILIN_TOPO
  Bodenbedeckung,BoFlaeche_Geometrie => BOUNDARY
END_MAP
```

Objekt	Beschreibung
<Table>_Boundary	Zusätzliches Objekt wie das Objekt mit dem Zentroid mit der Extension _Boundary. Das Objekt beinhaltet je eine Map mit den Komponenten der linken und rechten Fläche, falls die Flächen vorhanden sind. Beispiel: BoFlaeche ist das Objekt mit dem Zentroid. BoFlaeche_Boundary ist das Objekt mit den berechneten, gültigen Begrenzungslinien.

Komponente	Beschreibung
IN.GEOM(1)	Geometrie der Begrenzungslinie.
IN.LEFT(m)	Map mit den Komponenten des Objektes der linken Fläche. Nur vorhanden falls auch eine linke Fläche vorhanden ist.
IN.RIGHT(m)	Map mit den Komponenten des Objektes der rechten Fläche. Nur vorhanden falls auch eine rechte Fläche vorhanden ist.

BAD

Berechnete, ungültige Bergrenzungslinien werden zur Verfügung gestellt.

```
MAP ILIN_TOPO
  Bodenbedeckung,BoFlaeche_Geometrie => BAD
END_MAP
```

Objekt	Beschreibung
<Table>_Bad_Boundary	Zusätzliches Objekt wie das Objekt mit dem Zentroid mit der Extension _Bad_Boundary. Beispiel: BoFlaeche ist das Objekt mit dem Zentroid. BoFlaeche_Bad_Boundary ist das Objekt mit den berechneten, ungültigen Begrenzungslinien.

Komponente	Beschreibung
IN.GEOM(1)	Geometrie der ungültigen Begrenzungslinie.

Weiter Komponenten gemäss INTERLIS Datenmodell

Die restlichen Objektkomponenten sind abhängig von der dazugehörigen INTERLIS Tabelle (s.a. IN.MODEL, IN.TOPIC bzw. IN.TABLE). Alle INTERLIS Attribute werden als Komponenten des IN-Objekts mit dem gleichem Namen geliefert. Die INTERLIS Datentypen werden wie folgt auf ICS Datentypen abgebildet:

INTERLIS Datentyp	ICS Datentyp
IRANGE	int.
RRANGE	real.
Text	string.
GRADS	real.
DEGREES	real.
RADIANS	real.

ENUMERATION	int. string bei ILIN_PARAM.ENUM_TO_TEXT=ON
COORD2	point.
COORD3	point.
POLYLINE	line.
SURFACE	area. bei ILIN_PARAM.CALC_SURFACE=ON
AREA	point. Zentroid. area bei entsprechendem Eintrag in ILIN_TOPO.
-> (Referenz)	ilink. Referenzen von INTERLIS-Objekten auf andere INTERLIS-Objekte (z.B. Hoehen.Entstehung : -> HoehenNachfuehrung) sind als spezieller Datentyp ilink implementiert. INTERLIS-Referenzen können aufgelöst werden, indem deren Komponenten in iG/Script angesprochen werden (z.B. IN.Entstehung.Identifikator)

13.6. Exportierte Prozeduren und Methoden

Prozedur	<code>ILTOPO_OPEN [s input][[]]</code>
Beschreibung	Öffnet eine bestehende INTERLIS 1 Datei <input> und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'test.itf' ILTOPO_OPEN</code>
Prozedur	<code>ILTOPO_READ_OBJECT [][[b state]</code>
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten INTERLIS 1 Datei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>ILTOPO_READ_OBJECT [TRUE]</code>
Prozedur	<code>ILTOPO_CLOSE [][[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>ILTOPO_CLOSE</code>
Methode	<code>ILIN.GET_ILINK_KEY [ilink il] [s key]</code>
Beschreibung	Schlüssel aus Beziehungsattribut lesen.
Beispiel	<code>IN.Entstehung ILIN.GET_ILINK_KEY ['200015']</code>
Methode	<code>ILIN.GET_ILINK_TABLE [ilink il] [s table]</code>
Beschreibung	Tabellenname der referenzierten Tabelle aus Beziehungsattribut lesen.
Beispiel	<code>IN.Objekt ILIN.GET_ILINK_TABLE ['LFPNachfuehrung']</code>
Methode	<code>ILIN.GET_OBJECT_ILINK_LIST [m object, i linkdepthmax] [li list]</code>
Beschreibung	Liefert zu einem Input-Objekt - enthalten in der Map object - alle Objekte als Maps in einer Liste list zurück, auf die das Input-Objekt über Beziehungsattribute referenziert. Falls das Input-Objekt keine Beziehungsattribute aufweist, ist die Liste leer. Mit den Argument linkdepthmax kann die Tiefe der zu berücksichtigen Beziehungen definiert werden. Die Tiefe 0 liefert keine Objekte zurück. Die Tiefe 1 liefert alle Objekte zurück, die direkt vom Input-Objekt referenziert werden. Die Tiefe 2 liefert alle Objekte zurück, die direkt vom Input-Objekt referenziert werden und diejenigen,

die von diesen vom Input-Objekte referenzierten Objekte wiederum referenzieren. Und so weiter.

Beispiel	<pre>&IN 5 ILIN.GET_OBJECT_ILINK_LIST [list]</pre>
Methode	ILIN.COMPILE [s modelldatei] [s modell,b state]
Beschreibung	Modelldatei <modelldatei> (.ili) mit INTERLIS-Compiler compilieren. Falls das Modell keine Fehler enthält wird TRUE und der Name des Modells auf dem Stack geliefert, sonst FALSE. ILIN.COMPILE erzeugt ein Abbild des INTERLIS-Datenmodells in der vordefinierten Map ILIN_MODEL.
Beispiel	<pre>'td.ili' ILIN.COMPILE ['Grunddatensatz',TRUE]</pre>
Methode	ILIN.GET_MODEL [s modelldatei] [s modell,b state]
Beschreibung	Modellnamen aus der .ili Datei <modelldatei> lesen. Falls der Modellname gelesen werden konnte, wird TRUE und der Name des Modells auf dem Stack geliefert, sonst FALSE. ILIN.GET_MODEL ist für die Bestimmung des Modellnamens die effizientere Variante als ILIN.COMPILE. ILIN.GET_MODEL füllt jedoch im Gegensatz zu ILIN.COMPILE die Map ILIN_MODEL nicht.
Beispiel	<pre>'td.ili' ILIN.GET_MODEL ['Grunddatensatz',TRUE]</pre>
Methode	ILIN.SET_ALTRANGE [r minx,r miny,r minz,r maxx,r maxy,r maxz] []
Beschreibung	Alternativen Koordinatenbereich für Korrdinatenbereichstests festlegen. Der neue Koordinatenbereich übersteuert die aus dem INTERLIS-Datenmodell gelesenen Koordinatenbereiche.
Beispiel	<pre>600000.0 200000.0 500.0 650000.0 250000.0 600.0 ILIN.SET_ALTRANGE []</pre>

13.7. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von iltopo.mod
! gelesenen Objekte in der .log Datei an.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF      => \models\Grunddatensatz.ili
  TRACE             => OFF
  STATISTICS        => ON
END_MAP

MAP ILIN_TOPO
  Bodenbedeckung,BoFlaechе_Geometrie => AREA,BOUNDARY,BAD
  Liegenschaften,Liegenschaft_Geometrie => AREA,BOUNDARY,BAD
END_MAP
```

```

MAP INPUT_SOURCES
  I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
  I1 => DISPLAY_OBJECT1,IN
END_MAP

|INCL \script\iltopo.mod
|INCL \script\run1.prg

```

14. Modul ILTXTIN - INTERLIS 1 ohne Datenmodell lesen

14.1. Allgemeines

Der Modul ILTXTIN kann eine INTERLIS 1 .itf Datei *ohne* Datenmodell lesen. Das Lesen von .itf Dateien ohne die Angabe eines Datenmodells kann in einigen (seltenen) Fällen sinnvoll sein, z.B:

- Für Statistikkonfigurationen, welche generisch auf jeder .itf Datei funktionieren sollen (z.B. zum Zählen der Anzahl Objekte pro .itf Datei).
- Wenn der Input beim Lesen unter keinen Umständen (auch nicht bei fehlerhaften Daten) verändert werden darf, z.B. beim Splitten einer .itf Datei in mehrere Teile.

Der Modul wird mit:

```
|INCL \script\iltxtin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.



Der Modul ILTXTIN war in älteren Versionen der INTERLIS Tools als TEXTLIS verfügbar. Konfigurationen, welche noch auf TEXTLIS basieren, müssen daher auf ILTXTIN umgestellt werden.

14.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

14.3. Parametermap

Der Modul benötigt keine Parametermap.

14.4. Objektmodell

Der Modul liefert pro IN-Objekt folgende Komponenten:

Allgemeine Komponenten für jedes IN-Objekt

Komponente	req/opt	Beschreibung
IN.TXT(s)	r	Inhalt der aktuellen Zeile.

IN.LABEL(s)	r	Label der aktuellen Zeile. Unter Label versteht man die ersten 4 Zeichen der aktuellen Zeile (z.B. OBJE, TABL, TOPI, MODL, etc.)
IN.REST(s)	r	Rest der aktuellen Zeile ohne Label. Bemerkung allfällige Leerzeichen von IN.REST werden nicht eliminiert.
IN.LINE(i)	r	Zeilennummer der aktuellen Textzeile.

Zusätzliche Komponenten für IN.LABEL = 'SCNT'

Das IN-Objekt für SCNT enthält keine zusätzlichen Komponenten.

Zusätzliche Komponenten für IN.LABEL = 'HEAD'

Für alle Zeilen zwischen SCNT und //// werden Objekte vom Typ HEAD erzeugt. HEAD ist ein Pseudolabel, welches in der Inputdatei nicht existiert.

Zusätzliche Komponenten für IN.LABEL = '////'

Das IN-Objekt für //// enthält keine zusätzlichen Komponenten.

Zusätzliche Komponenten für IN.LABEL = 'MTID'

Dieses Label verfügt über keine zusätzlichen Komponenten.

Zusätzliche Komponenten für IN.LABEL = 'MODL'

Komponente	req/opt	Beschreibung
IN.MODEL(s)	r	Datenmodellname.

Zusätzliche Komponenten für IN.LABEL = 'TOPI'

Komponente	req/opt	Beschreibung
IN.MODEL(s)	r	Datenmodellname.
IN.TOPIC(s)	r	Aktueller Topicname.

Zusätzliche Komponenten für IN.LABEL = 'TABL'

Komponente	req/opt	Beschreibung
IN.MODEL(s)	r	Datenmodellname.
IN.TOPIC(s)	r	Aktueller Topicname.
IN.TABLE(s)	r	Aktueller Tablename.

Zusätzliche Komponenten für IN.LABEL = 'OBJE'

Das Label OBJE verfügt über keine zusätzlichen Komponenten.

Zusätzliche Komponenten für IN.LABEL = 'ETOP'

Dieses Label verfügt über keine zusätzlichen Komponenten.

Zusätzliche Komponenten für IN.LABEL = 'ETAB'

Dieses Label verfügt über keine zusätzlichen Komponenten.

Zusätzliche Komponenten für IN.LABEL = 'EMOD'

Dieses Label verfügt über keine zusätzlichen Komponenten.

Zusätzliche Komponenten für IN.LABEL = 'ENDE'

Dieses Label verfügt über keine zusätzlichen Komponenten.

14.5. Exportierte Prozeduren und Methoden

Prozedur	<code>ILTXTIN_OPEN [s input][]</code>
Beschreibung	Öffnet eine bestehende INTERLIS 1 Datei <input> und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'test.itf' ILTXTIN_OPEN</code>
Prozedur	<code>ILTXTIN_READ_OBJECT [][b state]</code>
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten INTERLIS 1 Datei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>ILTXTIN_READ_OBJECT [TRUE]</code>
Prozedur	<code>ILTXTIN_CLOSE [][]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>ILTXTIN_CLOSE</code>

14.6. Skriptbeispiel

```
! This configuration shows all objects
! read by ILTXTIN in the log file.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP INPUT_SOURCES
  I1 => ILTXTIN,OPT.input
END_MAP

MAP INOUT
  I1 => DISPLAY_OBJECT1,IN
END_MAP

|INCL \script\iltxtin.mod
|INCL \script\run1.prg
```

15. Modul LOGIN - ICS Logdateien lesen

15.1. Allgemeines

Mit dem Modul LOGIN können ICS Logdateien gelesen werden. Es kann verlangt werden, dass bestimmte Einträge in der .log Datei als Objekte geliefert werden (z.B. mit DISP angezeigte Linien und Flächen).

LOGIN wird mit:

```
| INCL \script\login.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

15.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

15.3. Parametermap LOGIN_PARAM

Folgende Parameter können in der Map LOGIN_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
PARSE_MESSAGE	r	ON oder OFF. Normale Meldungen in der .log Datei liefern. Es wird pro Meldungszeile ein IN-Objekt geliefert.
PARSE_ERROR	r	ON oder OFF. Fehlermeldungen als Objekt liefern.
PARSE_GEOMETRY	r	ON oder OFF. DISP-formatierte Geometrien vom Typ line oder area als Objekt liefern.
DEBUG	r	ON oder OFF. Debugmodus ein- bzw. ausschalten.

15.4. Objektmodell

Der Modul LOGIN liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.LINE(i)	r	Zeilennummer der aktuellen Logzeile. Die erste Zeile hat die Nummer 1.
IN.TYPE(s)	r	Type von IN.VALUE. Mögliche Werte sind: MESSAGE, ERROR, LINE und AREA. Welche Typen geliefert werden, hängt von den PARSE_* Parametern der LOGIN_PARAM Map ab.
IN.VALUE(o)	r	Wert des Objekts.

15.5. Exportierte Prozeduren und Methoden

Prozedur	LOGIN_OPEN ! [s input][]
Beschreibung	Öffnet die Logdatei <input>. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	'c:\test\test.log' LOGIN_OPEN
Prozedur	LOGIN_READ_OBJECT ! [][b state]
Beschreibung	Liest das nächste Objekt aus der aktuellen Logdatei. Das Objekt wird in der MAP IN zurückgegeben.
Beispiel	LOGIN_READ_OBJECT [TRUE]
Prozedur	LOGIN_CLOSE ! [][]

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel LOGIN_CLOSE

15.6. Skriptbeispiel

```
!+++++
!  
! Diese Konfiguration liest Objekte aus einer ICS .log Datei  
! und zeigt diese an.  
!  
  
|LICENSE \license\iltoolspro.lic  
|LICENSE \license\iltools.lic  
  
MAP CONFIG_PARAM  
  TYPE => LOGIN  
  VERSION => 1.7  
END_MAP  
  
!+++++  
!  
! user input  
!  
  
MAP USER_INPUT1  
  DIALOG => FILE  
  MESSAGE => 'Enter .log File'  
  FILE_FILTER => log  
  FILE_EXISTS => TRUE  
  OPT => input  
END_MAP  
  
!+++++  
!  
! parameters for DGEOM input module  
!  
  
MAP LOGIN_PARAM  
  PARSE_MESSAGE => ON ! deliver normal messages  
  PARSE_ERROR => ON ! deliver error messages  
  PARSE_GEOMETRY => ON ! parse geometries in DISP-format  
  DEBUG => OFF ! activate debugging mode  
END_MAP  
  
!+++++  
!  
! input sources  
!  
  
MAP INPUT_SOURCES  
  I1 => LOGIN,OPT.input  
END_MAP  
  
!+++++
```

```

!
! input/output mappings
!

MAP INOUT
  I1 => DISPLAY_OBJECT1,IN
END_MAP

!+++++
!
! macro declarations
!

MAP MACRO
END_MAP

|INCL \script\util.lib
|INCL \script\login.mod
|INCL \script\run1.prg

```

16. Modul MYSQLIN - MySQL lesen

16.1. Allgemeines

Mit dem Skriptmodul können Objekte aus einer MySQL-Datenbank via ODBC gelesen werden. Der Modul unterstützt speziell die OGC Option von MySQL für räumliche Daten.

MYSQLIN unterstützt die Geometry-Typen POINT, LINESTRING, POLYGON von MySQL nach der OGC Simple Feature Specification.

Der Modul wird mit:

```
|INCL \script\mysqlin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

16.2. Abhängigkeiten von anderen Modulen

Der Modul MYSQLIN ist eine Erweiterung des Moduls DBIN. Alle im Modul DBIN beschriebenen Anteile gelten auch für das Modul MYSQLIN. Ziehen Sie deshalb auch die Dokumentation des Modules DBIN bei.

16.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.

PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

16.4. Parametermap DBIN_PARAM

Folgende Parameter können in der Map DBIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
SELECT_PARSE	o	ON oder OFF, Default = ON. Definiert, ob ein eventuelles SQL-select Statement geparkt werden soll. Als Argument für für das Modul kann ein SQL-select Statement definiert werden. Als Default wird dieses select-Statement von geparkt , um spezielle Attribute zu erkennen, die vom ODBC-Treiber nicht unterstützt werden und speziell gelesen werden müssen. Manchmal können select-Statements (komplexe Joins) nicht erfolgreich geparkt werden. In solchen Fällen ist dieser Parameter auf OFF zu schalten. Sie müssen besorgt sein, dass das select-Statement von ODBC verarbeitet werden kann.

16.5. Parametermap MYSQLIN_PARAM

Folgende Parameter können in der Map MYSQLIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SPATIAL_PROCESS	o	ON oder OFF, Default = OFF. Definiert ob Spatial Geometrien gelesen werden sollen. Mit OFF werden die Geometrien nicht gelesen. Mit ON werden die Geometrien gelesen.

Für die Anwendung der Spatial Extension von MySQL ist die entsprechende Dokumentation von MySQL zu beachten.

16.6. Objektmodell

Der Modul liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN. TABLE(s)	r	Tablename des IN Objekts.
IN. <Geometry>(g)	o	Ein Geometrie-Attribut wird mit demselben Namen wie in der Datenbank geliefert. Der enthaltene Wert entspricht einer ICS-Geometrie vom Typ point, line oder area oder Listen von Geometrien.
IN. <Geometry>_OGC-CWKT_GEOMETRY(s)	o	Zusätzlich zu einem Geometrie-Attribut wird in einem Attribut mit dem Suffix _OGCWKT_GEOMETRY die OGC Geometrie als OGC-WKT-String geliefert (WKT: Well Known Text nach OGC Simple Feature Specification).

Alle weiteren Komponenten sind abhängig von der Tabellen-Definition in der Datenbank.

16.7. Exportierte Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie der Modul DBIN. Ziehen Sie deshalb die Dokumentation des Modules DBIN bei. Zusätzlich stellt der Modul MYSQLIN folgende Prozeduren und Methoden zur Verfügung.

Prozedur	MYSQLIN_OPEN [<i>s input</i>][<i>]</i>
Beschreibung	Öffnet eine Datenbank definiert mit <code>DB_PARAM.SOURCE</code> und liest Objekte von der Datenbank in Abhängigkeit von <code><input></code> . Für <code><input></code> können folgende Werte verwendet werden. *Liest alle Tabellen der Datenbank. <code><tablename></code> Liest die Records der definierten Tabelle. <code><tablename></code> , <code><sql-select-statement></code> Liest die Records der definierten Tabelle entsprechend dem SQL-select-Statement. Beinhaltet das SQL-select-Statement Hochkommas für Strings so sind die Hochkommas ' durch Anführungszeichen " zu ersetzen.Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>'CATEGORY,select * from CATEGORY where CNAME like "av%"' MYSQLIN_OPEN MAP INPUT_SOURCES I1 => MYSQLIN,CATEGORY,'select * from CATEGORY where CNAME like "av%"' END_MAP</pre>
Prozedur	MYSQLIN_READ_OBJECT [<i>][b state</i>]
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten Datenbank-Tabelle. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>MYSQLIN_READ_OBJECT [TRUE]</pre>
Prozedur	MYSQLIN_CLOSE [<i>][</i>]
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>MYSQLIN_CLOSE</pre>
Methode	OGC.WKT2GEOM [<i>s ogc-wkt-geometry</i>][<i>g li ics-geometry</i>]
Beschreibung	Übersetzt eine OGC-WKT-Geometrie als String in eine ICS-Geometrie.Folgende Typen werden zurückgegeben: POINT,LINESTRING,POLYGON,MULTIPOINT,MULTILINESTRING,MULTIPOLYGON,GEOMETRYCOLLECTION. (WKT: Well Known Text nach OGC Simple Feature Specification).
Beispiel	<pre>IN.Geometrie OGC.WKT2GEOM => VAR.GEOM</pre> <p>Folgende Konversionen werden durchgeführt:OGC-pointto pointOGC-linestringto lineOGC-polygonto areaOGC-multipointto list of pointsOGC-multilinestringto list of linesOGC-multipolygonto list of areasOGC-geometrycollectionto list of points und/oder lines und/oder areas</p>

16.8. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von mysqlin.mod
! gelesenen Objekte in der .log Datei an.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
```

```

DIALOG      => ODBC
OPT         => input
END_MAP

MAP DB_PARAM
SOURCE     => '' ! ODBC-Source
USER       => '' ! ODBC-User
PASSWD     => '' ! ODBC-Password
TRACE      => OFF
END_MAP

MAP DBIN_PARAM
STATISTICS      => ON
END_MAP

MAP MYSQLIN_PARAM
SPATIAL_PROCESS => ON
END_MAP

MAP INPUT_SOURCES
I1 => MYSQLIN,*
END_MAP

MAP INOUT
I1 => DISPLAY_IN0
END_MAP

| INCL \script\mysqlin.mod
| INCL \script\db2il\dbdisplay.out
| INCL \script\run1.prg

```

17. Modul ORAIN - Oracle Datenbank lesen

17.1. Allgemeines

Mit dem Skriptmodul können Objekte aus einer Oracle-Datenbank via ODBC gelesen werden. Der Modul unterstützt speziell die Oracle Option Spatial für räumliche Daten.

ORAIN unterstützt sämtliche Geometry-Typen von Oracle Spatial 9.2 und 10g. Diese sind POINT, LINE, POLYGON, MULTIPOINT, MULTILINE, MULTIPOLYGON und COLLECTION.

Der Modul wird mit:

```
| INCL \script\orain.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

17.2. Abhängigkeiten von anderen Modulen

Der Modul ORAIN ist eine Erweiterung des Moduls DBIN. Alle im Modul DBIN beschriebenen Anteile gelten auch für das Modul ORAIN. Ziehen Sie deshalb auch die Dokumentation des Modules DBIN bei.

17.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

17.4. Parametermap DBIN_PARAM

Folgende Parameter können in der Map DBIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
SELECT_PARSE	o	ON oder OFF, Default = ON. Definiert, ob ein eventuelles SQL-select Statement geparkt werden soll. Als Argument für für das Modul kann ein SQL-select Statement definiert werden. Als Default wird dieses select-Statement von geparkt , um spezielle Attribute zu erkennen, die vom ODBC-Treiber nicht unterstützt werden und speziell gelesen werden müssen. Manchmal können select-Statements (komplexe Joins) nicht erfolgreich geparkt werden. In solchen Fällen ist dieser Parameter auf OFF zu schalten. Sie müssen besorgt sein, dass das select-Statement von ODBC verarbeitet werden kann.

17.5. Parametermap ORAIN_PARAM

Folgende Parameter können in der Map ORAIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SPATIAL_PROCESS	o	ON oder OFF, Default = OFF. Definiert ob Oracle Spatial Geometrien gelesen werden sollen. Mit OFF werden die Geometrien nicht gelesen. Mit ON werden die Geometrien gelesen.
FENCE	o	string, Default = OFF. Wenn ein Fence nicht mit OPT.fence definiert ist, kann mit diesem Parameter ein Fence definiert werden. Der Wert kann in folgenden zwei Varianten gesetzt werden. Variante als Point-Array mit geschlossener Fläche in der Form $x1/y1, x2/y2, \dots$. Variante als ein SQL-Statement, das eine Fläche als Geometrie liefert. Beispiel: <code>select iltools_sdo_geometry_to_string(geom) as geom from filter where name like "name1"</code> . In der Variante als SQL-Statement muss die Geometrie wie im Beispiel mit der PL/SQL Funktion <code>ILTOOLS_SDO_GEOMETRY_TO_STRING</code> gelesen werden. Hochkommas ' im SQL-Statement

		müssen durch " ersetzt werden, weil in ICS das Hochkomma bereits als String-Delimitator verwendet wird.
FENCE_FILTER	o	string, Default = OFF. Definiert wie ein Fence als räumlicher Filter verwendet wird. Beispiel: <code>sdo_relate(%GEOMETRY%,%FENCE%, "mask=inside+coveredby querytype=WINDOW")="TRUE" ' . %GEOMETRY%</code> ist der Plazhalter für das Geometrie-Attribut des Objektes und wird durch das entsprechende Attribut ersetzt. <code>%FENCE%</code> ist der Plazhalter für die Fence-Geometrie und wird durch die Geometrie des Fences ersetzt. Hochkommas ' im SQL-Statement müssen durch " ersetzt werden, weil in ICS das Hochkomma bereits als String-Delimitator verwendet wird. Das Statement des Filters wird als where-clause dem select-Statement für die Abfrage der Datenbank-Tabelle zugefügt. Besitzt eine Datenbank-Tabelle mehrere Geometrie-Attribute, so werden die Filter für die einzelnen Geometrien mit einem or zusammengefügt.

Für die Anwendung der Oracle Option Spatial ist die entsprechende Dokumentation von Oracle zu beachten.

17.6. Objektmodell

Der Modul liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.TABLE(s)	r	Tablename des IN Objekts.
IN.<Geometry>(g)	o	Ein Geometrie-Attribut wird mit demselben Namen wie in der Datenbank geliefert. Der enthaltene Wert entspricht einer ICS-Geometrie vom Typ point, line oder area.
IN.<Geometry>_SDO_GEOMETRY(s)	o	Zusätzlich zu einem Geometrie-Attribut wird in einem Attribut mit dem Suffix <code>_SDO_GEOMETRY</code> die Oracle Spatial Geometrie als String geliefert.

Alle weiteren Komponenten sind abhängig von der Tabellen-Definition in der Datenbank.

17.7. Spezielles

Für das Lesen von Oracle Spatial sind folgende Punkte zu beachten.

Oracle Spatial lesen über ODBC mit Oracle PL/SQL

ODBC verarbeitet den Oracle Objekt-Typ `MDSYS.SDO_GEOMETRY` für die Geometrien nicht. Um die Geometrien über ODBC trotzdem lesen zu können, legt der Modul eine PL/SQL Funktion in Oracle an.

```
create function ILTOOLS_SDO_GEOMETRY_TO_STRING (geom MDSYS.SDO_GEOMETRY) RETURN CLOB
```

Diese Funktion wandelt eine Geometrie vom Typ `MDSYS.SDO_GEOMETRY` in einen String um. Die Funktion wird vom Modul beim Lesen der Geometrien wie folgt angewendet.

```
select ILTOOLS_SDO_GEOMETRY_TO_STRING(Geometrie) as Geometrie from Table
```

Die Funktion liefert die Geometrie als String, den der Modul dann in eine ICS-Geometrie umwandelt.

Der Oracle-User aus dem Daten gelesen werden sollen, muss deshalb die Berechtigung haben, eine Funktion anlegen zu können.

17.8. Exportierte Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie der Modul DBIN. Ziehen Sie deshalb die Dokumentation des Modules DBIN bei. Zusätzlich stellt der Modul ORAIN folgende Prozeduren und Methoden zur Verfügung.

Prozedur	ORAIN_OPEN [s input][]
Beschreibung	<p>Öffnet eine Datenbank definiert mit DB_PARAM.SOURCE und liest Objekte von der Datenbank in Abhängigkeit von <input>. Für <input> können folgende Werte verwendet werden.</p> <p>*</p> <p style="padding-left: 40px;">Liest alle Tabellen der Datenbank.</p> <p><tablename></p> <p style="padding-left: 40px;">Liest die Records der definierten Tabelle.</p> <p><tablename>,<sql-select-statement></p> <p style="padding-left: 40px;">Liest die Records der definierten Tabelle entsprechend dem SQL-select-Statement. Beinhaltet das SQL-select-Statement Hochkommas für Strings so sind die Hochkommas ' durch Anführungszeichen " zu ersetzen.</p> <p>Die Prozedur wird von RUN1 automatisch aufgerufen.</p>
Beispiel	<pre>'CATEGORY,select * from CATEGORY where CNAME like "av%" ' ORAIN_OPEN</pre> <pre>MAP INPUT_SOURCES I1 => ORAIN,CATEGORY,'select * from CATEGORY where CNAME like "av%" ' END_MAP</pre>
Prozedur	ORAIN_READ_OBJECT [] [b state]
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten Datenbank-Tabelle. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	ORAIN_READ_OBJECT [TRUE]
Prozedur	ORAIN_CLOSE [] []
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	ORAIN_CLOSE
Methode	ORACLE.SDO_GEOMETRY_TO_GEOM [s sdo-geometry][(i v alignment,)(i h alignment,)(r rotation,)(g l i geometry, i dimension, s type, b status]
Beschreibung	Übersetzt eine Oracle-Spatial Geometrie als String in eine ICS-Geometrie. Kan die Geometry übersetzt werden wird als Status TRUE zurückgegen, ansonsten FALSE. Je nach SDO-Type wird die Geometry als einzelne Geometry oder als Liste von Geometrien zurückgegeben. Beim GeoMedia-Typ gmpoint wird zusätzlich die Rotation geliefert. Beim GeoMedia-Typ gmtext wird zusätzlich die Rotation, der Text, das horizontale und das vertikale Alignment geliefert. Folgende Typen werden zurückgegeben: point,line,polygon,multipoint,multiline,multipolygon,collection,gmpoint,gmtext.

Beispiel

```

IF IN.Geometrie ORACLE.SDO_GEOMETRY_TO_GEOM THEN
=> VAR.TYPE
=> VAR.DIM
=> VAR.GEOM
IF    VAR.TYPE = 'gmpoint' THEN
=> VAR.ROT
ELSIF VAR.TYPE = 'gmtext' THEN
=> VAR.ROT
=> VAR.HALI
=> VAR.VALI
END_IF
END_IF

```

Folgende Konversionen werden durchgeführt:

SDO-point

to point

SDO-line

to line

SDO-polygon

to area

SDO-multipoint

to list of points

SDO-multiline

to list of lines

SDO-multipolygon

to list of areas

SDO-collection

to list of points und/oder lines und/oder areas

GEOMEDIA-SDO-point

to point and orientation

GEOMEDIA-SDO-text

to point and orientation, text, horizontal alignment, vertical alignment

17.9. Skriptbeispiel

```

! Diese ICS Konfiguration zeigt alle von orain.mod
! gelesenen Objekte in der .log Datei an.

```

```

|LICENSE \license\iltoolspro.lic

```

```

MAP USER_INPUT1
  DIALOG      => ODBC
  OPT         => input
END_MAP

```

```

MAP DB_PARAM
  SOURCE      => '' ! ODBC-Source
  USER        => '' ! ODBC-User
  PASSWD      => '' ! ODBC-Password

```

```

TRACE      => OFF
END_MAP

MAP ORAIN_PARAM
  STATISTICS      => ON
  SPATIAL_PROCESS => ON
END_MAP

MAP INPUT_SOURCES
  I1 => ORAIN,*
END_MAP

MAP INOUT
  I1 => DISPLAY_IN0
END_MAP

|INCL \script\orain.mod
|INCL \script\db2il\dbdisplay.out
|INCL \script\run1.prg

```

18. Modul PGRESIN - PostGreSQL/PostGIS Datenbank lesen

18.1. Allgemeines

Mit dem Skriptmodul können Objekte aus einer POstGreSQL/PostGIS-Datenbank via ODBC gelesen werden. Der Modul unterstützt speziell die PostGreSQL Option PostGIS für räumliche Daten.

PGRESIN unterstützt sämtliche Geometry-Typen von PostGIS nach der OGC Simple Feature Specification.

Der Modul wird mit:

```
|INCL \script\pgresin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

18.2. Abhängigkeiten von anderen Modulen

Der Modul PGRESIN ist eine Erweiterung des Moduls DBIN. Alle im Modul DBIN beschriebenen Anteile gelten auch für das Modul PGRESIN. Ziehen Sie deshalb auch die Dokumentation des Modules DBIN bei.

18.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.

USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

18.4. Parametermap DBIN_PARAM

Folgende Parameter können in der Map DBIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
SELECT_PARSE	o	ON oder OFF, Default = ON. Definiert, ob ein eventuelles SQL-select Statement geparkt werden soll. Als Argument für für das Modul kann ein SQL-select Statement definiert werden. Als Default wird dieses select-Statement von geparkt , um spezielle Attribute zu erkennen, die vom ODBC-Treiber nicht unterstützt werden und speziell gelesen werden müssen. Manchmal können select-Statements (komplexe Joins) nicht erfolgreich geparkt werden. In solchen Fällen ist dieser Parameter auf OFF zu schalten. Sie müssen besorgt sein, dass das select-Statement von ODBC verarbeitet werden kann.

18.5. Parametermap PGRESIN_PARAM

Folgende Parameter können in der Map PGRESIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SPATIAL_PROCESS	o	ON oder OFF, Default = OFF. Definiert ob Spatial Geometrien gelesen werden sollen. Mit OFF werden die Geometrien nicht gelesen. Mit ON werden die Geometrien gelesen.

Für die Anwendung der PostgreSQL Option PostGIS ist die entsprechende Dokumentation von PostGIS zu beachten.

18.6. Objektmodell

Der Modul liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.TABLE(s)	r	Tablename des IN Objekts.
IN.<Geometry>(g)	o	Ein Geometrie-Attribut wird mit demselben Namen wie in der Datenbank geliefert. Der enthaltene Wert entspricht einer ICS-Geometrie vom Typ point, line oder area oder Listen von Geometrien.
IN.<Geometry>_OGC- CWKT_GEOMETRY(s)	o	Zusätzlich zu einem Geometrie-Attribut wird in einem Attribut mit dem Suffix _OGCWT_GEOMETRY die PostGIS Geometrie

		als OCS-WKT-String geliefert (WKT: Well Known Text nach OGC Simple Feature Specification).
--	--	--

Alle weiteren Komponenten sind abhängig von der Tabellen-Definition in der Datenbank.

18.7. Exportierte Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie der Modul DBIN. Ziehen Sie deshalb die Dokumentation des Modules DBIN bei. Zusätzlich stellt der Modul PGRESIN folgende Prozeduren und Methoden zur Verfügung.

Prozedur	<code>PGRESIN_OPEN [s input][[]]</code>
Beschreibung	Öffnet eine Datenbank definiert mit <code>DB_PARAM.SOURCE</code> und liest Objekte von der Datenbank in Abhängigkeit von <code><input></code> . Für <code><input></code> können folgende Werte verwendet werden. *Liest alle Tabellen der Datenbank. <code><tablename></code> Liest die Records der definierten Tabelle. <code><tablename></code> , <code><sql-select-statement></code> Liest die Records der definierten Tabelle entsprechend dem SQL-select-Statement. Beinhaltet das SQL-select-Statement Hochkommas für Strings so sind die Hochkommas ' durch Anführungszeichen " zu ersetzen.Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>'CATEGORY,select * from CATEGORY where CNAME like "av%" PGRESIN_OPEN MAP INPUT_SOURCES I1 => PGRESIN,CATEGORY,'select * from CATEGORY where CNAME like "av%" END_MAP</pre>
Prozedur	<code>PGRESIN_READ_OBJECT [][b state]</code>
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten Datenbank-Tabelle. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>PGRESIN_READ_OBJECT [TRUE]</pre>
Prozedur	<code>PGRESIN_CLOSE [][[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>PGRESIN_CLOSE</pre>
Methode	<code>OGC.WKT2GEOM [s ogc-wkt-geometry][g li ics-geometry]</code>
Beschreibung	Übersetzt eine OGC-WKT-Geometrie als String in eine ICS-Geometrie.Folgende Typen werden zurückgegeben: POINT,LINESTRING,POLYGON,MULTIPOINT,MULTILINESTRING,MULTIPOLYGON,GEOMETRYCOLLECTION. (WKT: Well Known Text nach OGC Simple Feature Specification).
Beispiel	<pre>IN.Geometrie OGC.WKT2GEOM => VAR.GEOM</pre> <p>Folgende Konversionen werden durchgeführt:</p> <p>OGC-point to point</p> <p>OGC-linestring to line</p>

OGC-polygon

to area

OGC-multipoint

to list of points

OGC-multilinestring

to list of lines

OGC-multipolygon

to list of areas

OGC-geometrycollection

to list of points und/oder lines und/oder areas

18.8. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von pgresin.mod
! gelesenen Objekte in der .log Datei an.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG      => ODBC
  OPT         => input
END_MAP

MAP DB_PARAM
  SOURCE      => '' ! ODBC-Source
  USER        => '' ! ODBC-User
  PASSWD      => '' ! ODBC-Password
  TRACE       => OFF
END_MAP

MAP DBIN_PARAM
  STATISTICS  => ON
END_MAP

MAP PGRESIN_PARAM
  SPATIAL_PROCESS => ON
END_MAP

MAP INPUT_SOURCES
  I1 => PGRESIN,*
END_MAP

MAP INOUT
  I1 => DISPLAY_IN0
END_MAP

|INCL \script\pgresin.mod
|INCL \script\db2il\dbdisplay.out
|INCL \script\run1.prg
```

19. Modul SERIALIN - GeoShop Konfigurationsdateien lesen

19.1. Allgemeines

Mit dem Modul SERIALIN können GeoShop Konfigurationsdateien gelesen werden. Pro Konfigurationsdatei wird ein IN-Objekt geliefert.

SERIALIN wird mit:

```
|INCL \script\serialin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

19.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

19.3. Parametermap SERIALIN_PARAM

Folgende Parameter können in der Map SERIALIN_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	r	ON oder OFF. Objektstatistik am Ende der .log Datei ausgeben.
TRACE	o	Jedes gelesene Objekt in der .log Datei ausgeben.

19.4. Objektmodell

Der Modul SERIALIN liefert alle Komponenten gemäss der Struktur der gelesenen Konfigurationsdatei. Es werden keine allgemeinen Komponenten von SERIALIN geliefert.

19.5. Exportierte Prozeduren und Methoden

Prozedur	<code>SERIALIN_OPEN ! [s input_pattern][[]]</code>
Beschreibung	Öffnet alle Konfigurationsdateien <input_pattern>. Für <input_pattern> muss ein Pfad relativ zu GeoShop user_dir angegeben werden.
Beispiel	<code>'\users*.usr' SERIALIN_OPEN</code>
Prozedur	<code>SERIALIN_READ_OBJECT ! [[]b state]</code>
Beschreibung	Liest die nächste Konfigurationsdatei. Das Objekt wird in der MAP IN zurückgegeben.
Beispiel	<code>SERIALIN_READ_OBJECT [TRUE]</code>
Prozedur	<code>SERIALIN_CLOSE ! [[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>SERIALIN_CLOSE</code>

19.6. Skriptbeispiel

```
!=====  
! Display all GeoShop user names.  
!=====  
  
!=====  
! parameter maps for input modules  
!=====  
  
MAP SERIALIN_PARAM  
    TRACE      => OFF  
    STATISTICS => ON  
END_MAP  
  
!=====  
! input sources  
!=====  
  
MAP INPUT_SOURCES  
    I1 => SERIALIN, \users\*.usr  
END_MAP  
  
!=====  
! classification  
!=====  
  
MAP INOUT  
    I1 => DIN  
END_MAP  
  
!=====  
! macros  
!=====  
  
MAP MACRO ! macros  
    DIN => DISPLAY_OBJECT1, IN.name  
END_MAP  
  
!=====  
! includes  
!=====  
  
| INCL \script\util.lib  
| INCL \script\serialin.mod  
| INCL \script\run1.prg  
  
!=====  
! end of file  
!=====
```

20. Modul SHPIN - ESRI Shapefile lesen

20.1. Allgemeines

Mit dem Modul SHPIN können Objekte aus ESRI Shapefile Dateien gelesen werden.

SHPIN wird mit:

```
| INCL \script\shpin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

20.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

20.3. Parametermap SHPIN_PARAM

Folgende Parameter können in der Map SHPIN_PARAM für den Modul SHPIN gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	r	ON oder OFF. Objektstatistik am Ende der .log Datei ausgeben.
DEBUG	r	ON oder OFF. Debugmodus ein- oder ausschalten (Default = OFF).
DBF_DOS	o	ON oder OFF. Falls ON gesetzt wird, werden DBF-Dateien im DOS-Zeichensatz gelesen. Sonst wird der Windowszeichensatz gelesen.
READ_M	o	ON oder OFF. Falls ON gesetzt wird, werden die Measurement-Werte als Z-Koordinaten gelesen, falls vorhanden. Eventuelle Z-Koordinaten werden nicht gelesen. Measurement-Werte können in folgenden Shape Typen vorkommen: POINTZ, POINTM, MULTI-POINTZ, MULTIPPOINTM, POLYLINEZ, POLYLINEM, POLYGONZ, POLYGONM.

20.4. Objektmodell

Der Modul SHPIN liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.FILE(s)	r	Name der Inputdatei ohne Verzeichnis und ohne Endung.
IN.TYPE(s)	r	Typ des Objekts. Mögliche Werte für IN.TYPE sind: NULL, POINT, MULTIPPOINT, POLYLINE, POLYGON.
IN.SHPTYPE(s)	r	Shapetyp des Objekts. Mögliche Werte für IN.SHPTYPE sind: NULLSHP, POINT, POINTZ, POINTM, MULTIPPOINT, MULTIPPOINTZ, MULTIPPOINTM, POLYLINE, POLYLINEZ, POLYLINEM, POLYGON, POLYGONZ, POLYGONM.
IN.DIMENSION(s)	r	Dimension der Objektgeometrie. Mögliche Werte sind: 2D oder 3D.
IN.FEATUREID(i)	r	Featureid.
IN.MINX(r), IN.MI- NY(r) IN.MAXX(r), IN.MAXY(r)	r	XY-Rangbox um die Objektgeometrie IN.GEOM.
IN.MINZ(r), IN.MINZ(r)	o	Z-Range der Objektgeometrie IN.GEOM, falls Z vorhanden ist.

IN.MINM(r), IN.MINM(r)	o	M-Range der Objektgeometrie IN.GEOM, falls M vorhanden ist.
IN.GEOM(p,l)	r	Objektgeometrie. Bemerkungen: Die Geometrie wird beim Typ POINT als Point zurückgegeben. Die Geometrie wird bei den Typen MULTIPOINT, POLYLINE und POLYGON als Liste zurückgegeben, auch wenn die Geometrie nur aus einem Teil besteht. Objekte vom Typ NULL haben keine Geometrie.

Neben dem IN-Objekt liefert der Modul SHPIN alle Attribute des aktuellen Objekts in der Map SHPIN_REC.

20.5. Exportierte Prozeduren und Methoden

Prozedur	<code>SHPIN_OPEN ! [s input][]</code>
Beschreibung	Öffnet die SHP Datei <input>. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'c:\test\test.shp' SHPIN_OPEN</code>
Prozedur	<code>SHPIN_READ_OBJECT ! [][b state]</code>
Beschreibung	Liest das nächste Objekt aus der aktuellen SHP-Inputdatei. Das Objekt wird in der MAP IN bzw. SHPIN_REC zurückgegeben. Falls Der Typ des gelesenen Objektes wird mit der Komponente IN.TYPE angegeben.
Beispiel	<code>SHPIN_READ_OBJECT [TRUE]</code>
Prozedur	<code>SHPIN_CLOSE ! [][]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>SHPIN_CLOSE</code>

20.6. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von shpin.mod
! gelesenen Objekte in der .log Datei an.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .shp Input File'
  FILE_FILTER => shp
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP SHPIN_PARAM
  STATISTICS => ON
  DEBUG      => OFF
END_MAP

MAP INPUT_SOURCES
  I1 => SHPIN,OPT.input
```

```

END_MAP

MAP INOUT
  I1 => DISPLAY_OBJECT1, IN, DISPLAY_OBJECT1, SHPIN_REC
END_MAP

| INCL \script\util.lib
| INCL \script\shpin.mod
| INCL \script\run1.prg

```

21. Modul SQLITEIN - SQLite-Datenbank lesen

21.1. Allgemeines

Mit dem Skriptmodul können Objekte aus einer SQLite-Datenbank gelesen werden.

Insbesondere werden die Spezifikationen von **OGC Geopackages** (Vektor-Geometrien) unterstützt.

Der Modul wird mit:

```
| INCL \script\sqlitein.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

21.2. Parametermap SQLITEIN_PARAM

Folgende Parameter können in der Map SQLITEIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.

21.3. Objektmodell

Der Modul liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.TABLE(s)	r	Tablename des IN Objekts.

Alle weiteren Komponenten sind abhängig von der Tabellen-Definition in der Datenbank.

21.4. Datenbankmodell

Die Methode `DB.GET_INFO` liefert das Datenmodell der Datenbank in einer Objektstruktur. Auf die einzelnen Komponenten dieser Objektstruktur kann zugegriffen werden.

Anwendung der Methode `DB.GET_INFO`:

```

MAP DB_MODEL
END_MAP

&DB_MODEL SQLite.GET_INFO

```

Objektstruktur Datenmodell:

```

DB_MODEL (m)
  DB_NAME      -> (s) <Name>
  DB_USER      -> (s) <DB-User>
  DB_PASSWORD  -> (s) <DB-Password>
  PRODUKT      -> (s) <Produkt>
  VERSION      -> (s) <Version>
  TABLES      -> (m)
  :
  <Tablename> -> (m)
    TABLENAME -> (s) <Tablename>
    TABLETYPE -> (s) <Tabletype>
    COLUMNS   -> (m)
    :
    <Columnname> -> (m)
      NAME      -> (s) <Name>
      TYPENAME  -> (s) <Typename>
      NULLABLE  -> (b) <Nullable>
      PRIMARYKEY -> (b) <Primary Key>

```

Zugriffsbeispiele Objektstruktur Datenmodell:

```

! Display User,Product,Version
DISPLAY DB_MODEL

! Display Tables
DISPLAY DB_MODEL.TABLES

! Display Table MyTable
DISPLAY DB_MODEL.TABLES.MyTable

! Display Columns of Table MyTable
DISPLAY DB_MODEL.TABLES.MyTable.COLUMNS

! Display Column MSLINK of Table MyTable
DISPLAY DB_MODEL.TABLES.MyTable.COLUMNS.MyColumn

! Display Typename of Column MyColumn of Table MyTable
DISPLAY DB_MODEL.TABLES.MyTable.COLUMNS.MyColumn.TYPENAME

```

21.5. Exportierte Prozeduren und Methoden

Prozedur	SQLITEIN_OPEN [s input][[]]
Beschreibung	Öffnet eine Datenbank und liest Objekte von der Datenbank in Abhängigkeit von <input>.
	Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	OPT.input SQLITEIN_OPEN
Prozedur	SQLITEIN_READ_OBJECT [][b state]
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten Datenbank. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	SQLITE_READ_OBJECT [TRUE]

Prozedur	SQLITEIN_CLOSE [[]]
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	SQLITEIN_CLOSE

21.6. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von SQLITEIN
! gelesenen Objekte in der .log Datei an.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter SQLITE Input File'
  FILE_FILTER => gpkg;db
  FILE_EXISTS => TRUE
  OPT         => input
END_MAP

MAP SQLITEIN_PARAM
  STATISTICS  => ON
END_MAP

MAP INPUT_SOURCES
  I1 => SQLITEIN,OPT.input
END_MAP

MAP INOUT
  I1 => DISPLAY_OBJECT1,IN
END_MAP

|INCL \script\sqlitein.mod
|INCL \script\run1.prg
```

22. Modul TXTIN - Textdateien lesen

22.1. Allgemeines

Mit dem Modul TXTIN können Textdateien zeilenweise gelesen werden. Es kann verlangt werden, dass eine Textzeile in einzelne Felder unterteilt wird.

TXTIN wird mit:

```
|INCL \script\txtin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

22.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

22.3. Parametermap TXTIN_PARAM

Folgende Parameter können in der Map TXTIN_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	r	ON oder OFF. Objektstatistik am Ende der .log Datei ausgeben.
DEBUG	r	ON oder OFF. Debugmodus ein- oder ausschalten (Default = OFF).
FIRST_LINE	o	ATTRIBUTE_NAMES oder OFF, Default=OFF. Falls die erste Zeile der Textdatei die Feldnamen enthält, kann man mit ATTRIBUTE_NAMES verlangen, dass diese entsprechend interpretiert wird.
SEPARATOR	o	<CHAR>, Default=' '. Separatorzeichen zwischen den einzelnen Feldern der Textzeile. Der Wert kann als Buchstaben oder als ASCII Code (z.B. 9 für Tabulator) angegeben werden. Falls als Wert ' ' angegeben wird, wird die aktuelle Zeile nicht in Felder unterteilt.
TRACE	o	Jedes gelesene Objekt in der .log Datei ausgeben.

22.4. Objektmodell

Der Modul TXTIN liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.LINE(i)	r	Zeilennummer der aktuellen Textzeile. Die erste Zeile hat die Nummer 1.
IN.TXT(s)	r	Inhalt der aktuellen Textzeile.

Falls der Parameter SEPARATOR definiert wurde, wird der Zeileninhalt in Felder aufgeteilt. Falls ausserdem FIRST_LINE = ATTRIBUTE_NAMES definiert wurde, wird jeder Komponente der Namen des dazugehörigen Feldnamen aus der 1. Zeile zugewiesen. Falls FIRST_LINE fehlt werden die Komponenten mit A1 .. An benannt. Alle so erzeugten Komponenten sind vom Typ STRING.

22.5. Exportierte Prozeduren und Methoden

Prozedur	<code>TXTIN_OPEN ! [s input][]</code>
Beschreibung	Öffnet die Textdatei <input>. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'c:\test\test.txt' TXTIN_OPEN</code>
Prozedur	<code>TXTIN_READ_OBJECT ! [][b state]</code>
Beschreibung	Liest das nächste Objekt aus der aktuellen Textdatei. Das Objekt wird in der MAP IN zurückgegeben.
Beispiel	<code>TXTIN_READ_OBJECT [TRUE]</code>
Prozedur	<code>TXTIN_CLOSE ! [][]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>TXTIN_CLOSE</code>

22.6. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von txtin.mod
! gelesenen Objekte in der .log Datei an.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .txt Input File'
  FILE_FILTER => txt
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP TXTIN_PARAM
  FIRST_LINE => ATTRIBUTE_NAMES
  SEPARATOR => 9
  STATISTICS => ON
  DEBUG => OFF
END_MAP

MAP INPUT_SOURCES
  I1 => TXTIN,OPT.input
END_MAP

MAP INOUT
  I1 => DISPLAY_OBJECT1,IN
END_MAP

|INCL \script\util.lib
|INCL \script\txtin.mod
|INCL \script\run1.prg
```

23. Modul XLSIN - MS EXCEL lesen

23.1. Allgemeines

Mit dem Skriptmodul können Objekte aus einer Excel File gelesen werden.

Der Modul wird mit:

```
|INCL \script\xlsin.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

23.2. Abhängigkeiten von anderen Modulen

Der Modul importiert die Klasse DB. Es stehen daher auch alle Methoden der Klasse DB zur Verfügung (s.a. iG/Script Benutzer- und Referenzhandbuch).

23.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

23.4. Parametermap DBIN_PARAM

Folgende Parameter können in der Map DBIN_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
SELECT_PARSE	o	ON oder OFF, Default = ON. Definiert, ob ein eventuelles SQL-select Statement geparkt werden soll. Als Argument für für das Modul kann ein SQL-select Statement definiert werden. Als Default wird dieses select-Statement von geparkt , um spezielle Attribute zu erkennen, die vom ODBC-Treiber nicht unterstützt werden und speziell gelesen werden müssen. Manchmal können select-Statements (komplexe Joins) nicht erfolgreich geparkt werden. In solchen Fällen ist dieser Parameter auf OFF zu schalten. Sie müssen besorgt sein, dass das select-Statement von ODBC verarbeitet werden kann.

23.5. Parametermap XLSIN_PARAM

In dieser Parameter Map sind zur Zeit keine Parameter definierbar.

Parameter	req/opt	Beschreibung
-----------	---------	--------------

23.6. Objektmodell

Der Modul liefert pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.TABLE(s)	r	Tablename des IN Objekts.
IN.ROW(i)	r	Zeile des IN Objekts.

Alle weiteren Komponenten sind abhängig von der Tabellen-Definition in der Datenbank.

23.7. EXCEL Tabelle vorbereiten

Tabelle gesamthaft lesen

Wenn der Modul eine Excel-Tabelle vollständig lesen soll, muss die Excel-Tabelle nach EXCEL-ODBC-Konvention wie folgt angesprochen werden:

```
[<table>$]
```

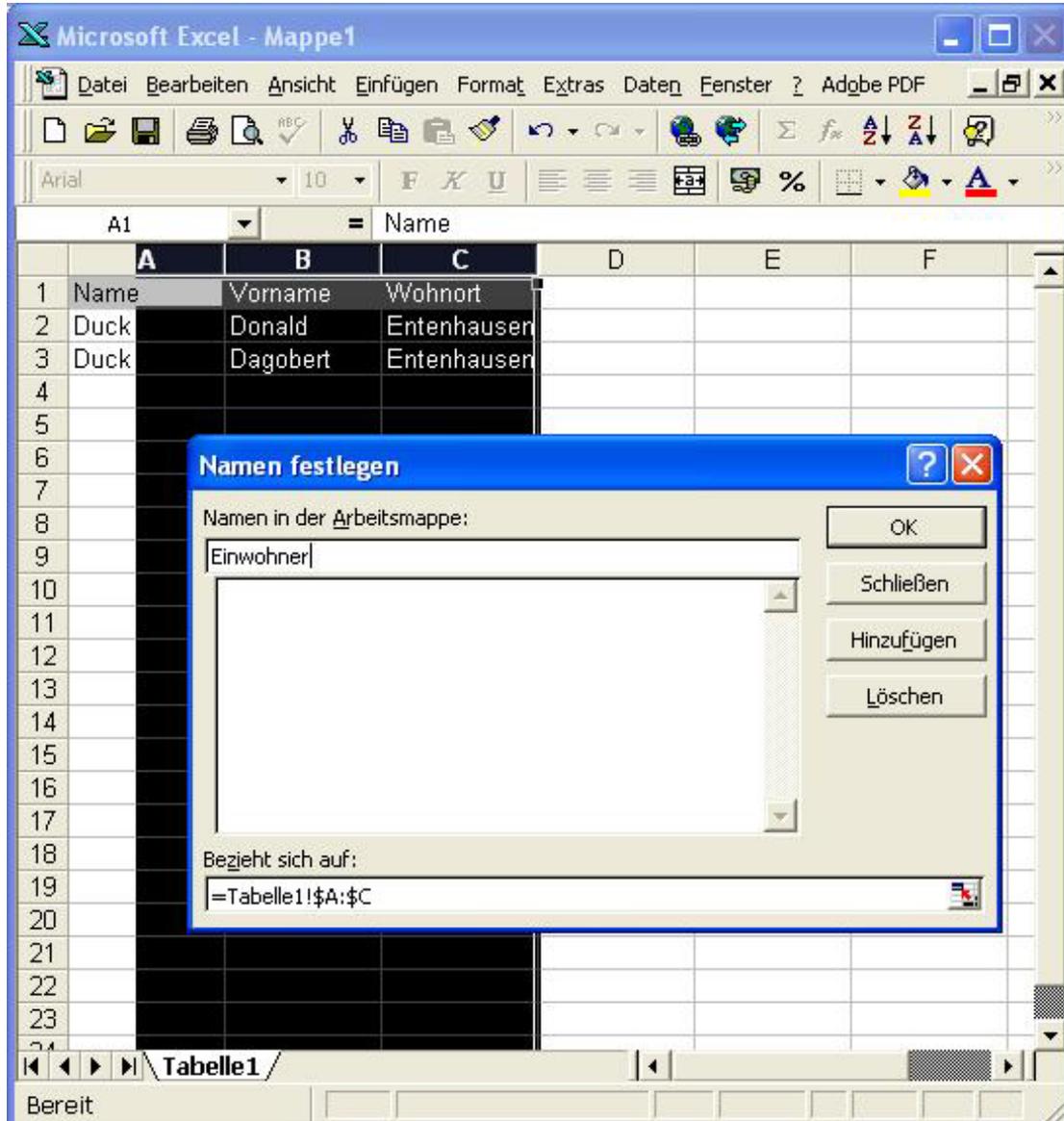
Beispiel:

```
MAP INPUT_SOURCES
  I1 => XLSIN,[Tabelle1$]
END_MAP
```

Tabelle tabellarisch lesen

Damit der Modul Daten tabellarisch - Teile einer Tabelle - aus einer Excel-Tabelle lesen kann, ist die Excel Tabelle wie folgt unter Excel vorzubereiten.

Abbildung A.1.



Spaltennamen

In der ersten Zeile sind die Spaltennamen einzutragen.

Tabellennamen

Die zu exportierenden Spalten sind zu markieren.

Über Einfügen > Namen > Festlegen ist dem markierten Bereich ein Namen zu vergeben.

Der markierte Bereiche wird über diesen Namen vom Modul XLSIN angesprochen.

z.B.

```
MAP INPUT_SOURCES
  I1 => XLSIN, Einwohner
END_MAP
```

23.8. Datenbankmodell

Die Methode `DB.GET_INFO` liefert das Datenmodell der Datenbank in einer Objektstruktur. Auf die einzelnen Komponenten dieser Objektstruktur kann zugegriffen werden.

Anwendung der Methode `DB.GET_INFO`:

```
MAP DB_MODEL
END_MAP

&DB_MODEL DB.GET_INFO
```

Objektstruktur Datenmodell:

```
DB_MODEL (m)
  DB_NAME      -> (s) <Name>
  DB_USER      -> (s) <DB-User>
  DB_PASSWORD  -> (s) <DB-Password>
  PRODUKT      -> (s) <Produkt>
  VERSION      -> (s) <Version>
  TABLES      -> (m)
    :
    <Tablename> -> (m)
      TABLENAME      -> (s) <Tablename>
      TABLEQUALIFIER -> (s) <Tabelqualifier>
      TABLEOWNER     -> (s) <Tableowner>
      TABLETYPE      -> (s) <Tabletype>
      COLUMNS        -> (m)
        :
        <Columnname> -> (m)
          NAME          -> (s) <Name>
          TYPENAME      -> (s) <Typename>
          DATATYPE      -> (i) <Datatype>
          LENGTH        -> (i) <Length>
          PRECISION     -> (i) <Precision>
          SCALE         -> (i) <Scale>
          RADIX         -> (i) <Radix>
          NULLABLE      -> (b) <Nullable>
          REMARKS       -> (s) <Remarks>
```

Zugriffsbeispiele Objektstruktur Datenmodell:

```
! Display User, Product, Version
DISPLAY DB_MODEL

! Display Tables
DISPLAY DB_MODEL.TABLES

! Display Table Category
DISPLAY DB_MODEL.TABLES.CATEGORY

! Display Columns of Table Category
DISPLAY DB_MODEL.TABLES.CATEGORY.COLUMNS

! Display Column MSLINK of Table Category
```

```
DISPLAY DB_MODEL.TABLES.CATEGORY.COLUMNS.MSLINK
```

```
! Display Typename of Column MSLINK of Table Category
DISPLAY DB_MODEL.TABLES.CATEGORY.COLUMNS.MSLINK.TYPENAME
```

23.9. Exportierte Prozeduren und Methoden

Prozedur	XLSIN_OPEN [<i>s input</i>][<i>]</i>
Beschreibung	<p>Öffnet ein Excel Tabelle. Für <input> können folgende Werte verwendet werden.</p> <p>*</p> <p style="padding-left: 40px;">Liest alle Tabellen der Datenbank.</p> <p><tablename></p> <p style="padding-left: 40px;">Liest die Records der Tabelle <tablename>.</p> <p><tablename>,<sql-select-statement></p> <p style="padding-left: 40px;">Liest die Records der Tabelle <tablename> entsprechend dem SQL-select-Statement. Beinhaltet das SQL-select-Statement Hochkommas für Strings so sind die Hochkommas ' durch Anführungszeichen " zu ersetzen.</p> <p>Die Prozedur wird von RUN1 automatisch aufgerufen.</p>
Beispiel	<pre>'CATEGORY,select * from CATEGORY where CNAME like "av%" ' XLSIN_OPEN</pre> <pre>MAP INPUT_SOURCES I1 => XLSIN,CATEGORY,'select * from CATEGORY where CNAME like "av%" ' END_MAP</pre>
Prozedur	XLSIN_READ_OBJECT [<i>]</i> [<i>b state</i>]
Beschreibung	Liest das nächste IN-Objekt aus der geöffneten Excel-Tabelle. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>XLSIN_READ_OBJECT [TRUE]</pre>
Prozedur	XLSIN_CLOSE [<i>]</i> [<i>]</i>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>XLSIN_CLOSE</pre>

23.10. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von XLSIN
! gelesenen Objekte in der .log Datei an.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter Excel Input File'
  FILE_FILTER => xls
  FILE_EXISTS => TRUE
  OPT        => input
```

```

END_MAP

MAP DB_PARAM
END_MAP

MAP DBIN_PARAM
  STATISTICS    => ON
END_MAP

MAP XLSIN_PARAM
END_MAP

MAP INPUT_SOURCES
  I1 => XLSIN,*
END_MAP

MAP INOUT
  I1 => DISPLAY_OBJECT1,IN
END_MAP

|INCL \script\xlsin.mod
|INCL \script\run1.prg

```

24. Modul XMLSAX - XML lesen

24.1. Allgemeines

Mit dem Modul XMLSAX können XML-Dateien gelesen werden. Das Modul basiert auf dem im Apache Projekt XERCES implementierten SAX Parser.

XMLSAX wird mit:

```
|INCL \script\xlsax.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

24.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

24.3. Parametermap XMLSAX_PARAM

Folgende Parameter können in der Map XMLSAX_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF. Objektstatistik am Ende der .log Datei ausgeben.
ELEMENT_LEVEL	o	<level> oder OFF. Definiert von welcher Verschachtelungstiefe (Level) die XML-Elemente geliefert werden sollen. z.B. Für INTERLIS XTF-Files ist der Level=4 sinnvoll, da damit die Instanzen der Klassen geliefert werden <TRANSFER><DATASECTION><Topic><Class> . Mit OFF werden alle Elemente aller Level geliefert. Dies eignet sich nicht bei sehr grossen XML Dateien. z.B würde bei einem INTERLIS-XTF-File unter allen Elementen auch der

		ganze Inhalt im Element <TRANSFER> geliefert. Dies könnte bei einem sehr grossen File zu einem Abbruch wegen zuwenig Memory führen.
ELEMENT_PARENT_INFO_LEVEL	o	<level> oder OFF. Bis zu welcher Verschachtelungstiefe (Level) Informationen zu Parentelementen eines Elementes geliefert werden sollen.

24.4. Objektmodell

Der Modul XMLSAX liefert pro IN-Objekt folgende Systemkomponenten:

Komponenten	req/opt	Beschreibung
IN.XML_LINE(i)	r	Zeilennummer der aktuellen Elementes
IN.XML_COLUMN(i)	r	Kolonnennummer des aktuellen Elementes
IN.XML_LEVEL(i)	r	Verschachtelungstiefe (Level) des aktuellen Elementes
IN.XML_ELEMENT_LEVEL_*(s)	o	Namen der Parentelemente pro Verschachtelungstiefe (Level) des aktuellen Elementes. Nur wenn Parameter ELEMENT_PARENT_INFO_LEVEL gesetzt ist.
IN.XML_ELEMENT_PARENT(s)	r	Namen des Parentelementes des aktuellen Elementes
IN.XML_ELEMENT(s)	r	Namen des aktuellen Elementes
IN.DATA(s)	o	Den Datenwert des aktuellen Elementes.
IN.<attribut>(s)	o	Attribute des aktuellen Elementes. Der Wert ist immer ein String.
IN.<element>(m li)	o	Childelemente des aktuellen Elementes. Ist das Childelement nur einmal vorhanden, so ist der Wert vom Typ MAP. Ist das Childelement mehrmals vorhanden, so ist der Wert vom Typ LIST mit einer MAP pro Child. Pro Child sind in der Map wieder die Komponenten des Objektmodelles vorhanden.

24.5. Objektmodell bei einem SAX-Ereignis

Das Modul liest das XML File als SAX Parser. Dabei werden vom SAX Parser Ereignisse (Events) gemeldet. Für die Events kann optional eine Prozedur gesetzt werden (siehe später). Die Prozedur erhält die Ereignisse in der Form des IN-Objektes geliefert. Das IN-Objekt weist folgende Komponenten auf.

Komponenten	req/opt	Beschreibung
IN.XML_LINE(i)	r	Zeilennummer des aktuellen Events
IN.XML_COLUMN(i)	r	Kolonnennummer des aktuellen Events
IN.XML_LEVEL(i)	r	Verschachtelungstiefe (Level) des aktuellen Events
IN.XML_SAXEVENT(s)	r	Name des aktuellen Events. Unterstützt werden zur Zeit die Events startElement, endElement, characters.
IN.XML_ELEMENT(s)	o	Namen des aktuellen Elementes. Nur wenn ein Element am Ereignis beteiligt ist. Beispiel Ereignis startElement, endElement
IN.<attribut>(s)	o	Attribute des aktuellen Elementes. Nur wenn das aktuelle Element des Ereignisses Attribute aufweist. Beispiel Ereignis startElement .

2. Modul ARCGISOUT - ESRI Geodatabase schreiben

2.1. Allgemeines

Mit dem Modul können Objekte in eine ESRI-Geodatabase geschrieben werden. Unterstützt werden:

- ESRI SDE Geodatabase (Oracle)
- ESRI Personal Geodatabase (ACCESS)
- ESRI File Geodatabase

Der Modul beinhaltet folgende Besonderheiten:

- Schreibt Daten mit Geometrien in eine Geodatabase.
- Legt Tabellen für die Daten und die Spatial Indexe entsprechende der ESRI Geodatabase an.
- Füllt das Geodatabase Repository mit den notwendigen Definitionen.
- Die Geodatabase kann nach dem Schreiben der Daten direkt mit den ESRI-Anwendungen weiterbearbeitet werden.

Der Modul wird mit:

```
| INCL \script\arccgisout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

2.2. ESRI Lizenz

Der Modul verwendet das ESRI ArcObjects API. Für die Anwendung des Modules ist deshalb eine ESRI Lizenz notwendig. Folgende minimalen ESRI Lizenzen sind notwendig.

ESRI Geodatabase	ESRI minimal Lizenz: Modell kreieren	ESRI minimal Lizenz: Daten schreiben
SDE	ArcEditor	ArcGIS Engine Runtime
Personal	ArcGIS Engine Runtime	ArcGIS Engine Runtime
File	ArcGIS Engine Runtime	ArcGIS Engine Runtime

Wenn Sie zum Beispiel eine SDE Geodatabase bearbeiten möchten, benötigen Sie zum Kreieren des Modelles in SDE die ArcEditor Lizenz.. Ist das Modell in der SDE Geodatabase einmal angelegt, benötigen Sie zum Schreiben der Daten nach SDE nur noch die ArcGIS Engine Runtime Lizenz.

2.3. Parametermap ARCGISOUT_PARAM

Folgende Parameter können in der Map ARCGISOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
-----------	---------	--------------

SRID	r	<STRING>. Der Name des ArcGIS Spatial Reference System. Es muss der Name eines ArcGIS bekannten Projected Coordinate Systems sein. Beispiele: 'CH1903 LV03', 'CH1903+ LV95'
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltet werden. Sie dazu mehr unter dem Kapitel Datasets im Module DBOUT.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
SPATI- AL_GEOM_CLEAN	o	ON oder OFF, Default = OFF. Bereinigt die Geometry mit ArcObject-Methode ITopologicalOperator.Simplify.
COMPRESS	o	ON oder OFF, Default = OFF. Soll die SDE Datenbank nach dem import komprimiert werden. Nur bei SDE-Datenbanken anwendbar.

2.4. ArcGIS SDE Connect

Der Connect zu einer ArcGIS SDE Datenbank wird unterschieden zwischen einem

- Direct Connect (ab 10.1 Standard)

und einem

- SDE Service Connect (bis 10.0 Standard)

Bis und mit SDE Version 10.0 war der Connect über einen SDE Service der Standard. Ab SDE Version 10.1 ist Direct Connect der Standard. Direct Connect benötigt keinen SDE Service, sondern verwendet die Möglichkeiten der Datenbank für einen Connect.

Bis SDE Version 10.0 konnte der SDE Service über das Postinstallations-Programm von SDE interaktiv eingerichtet werden. Ab SDE Version 10.1 muss der SDE Service manuell eingerichtet werden.

Entsprechend der Verbindung über Direct Connect oder SDE Service sind die Connect-Parameter etwas anders. Nachfolgend werden Beispiele für die Verbindung zweier geläufigen Datenbanken aufgeführt.

Datenbank	Connect	Connect Parameter Interaktiv	Connect Parameter Batch OPT.input/OPT.output
Oracle	Direct Connect	Server: '' Leer. Instance: sde:<Oracle-Client-Value> sde: mit Oracle-Client-Version je nach Client Version: Oracle (für 8i), Oracle9i, Oracle10g, Oracle11g Database: <Database> (Optional) Datenbank des Users. User: <User> Datenbank User Password: <Password>@<Oracle-Service> Password des Users und Oracle Service definiert für Client	Parameter ,<Instance>,<Database>,<User>,<Password>@<Oracle-Service>,<Version> Beispiel ,sde:Oracle11g,,test,info-grips@ORCL,

		Version: <Version> (Optional) Leer oder Default sde.DEFAULT.	
	SDE Service	Server: <Server> Server mit SDE Service Instance: <SDE Service Port> Port des SDE Services Database: <Database> (Optional) Datenbank des Users. User: <User> Datenbank User Password: <Password> Password des Users und Oracle Service definiert für Client Version: <Version> (Optional) Leer oder Default sde.DEFAULT.	Parameter <Server>,<Instance>,<Data- base>,<User>,<Pass- word>,<Version> Beispiel Server,5151,,test,infogrips,
SQL Server	Direct Connect	Server: <Server> Server mit SDE Datenbank. Instance: sde:sqlserver:<SQL Server Instance>> sde:sqlserver mit SQL Server Instance Database: <Database> (Optional) Datenbank des Users. User: <User> Datenbank User Password: <Password> Password des Users Version: <Version> (Optional) Leer oder Default dbo.DEFAULT.	Parameter <Server>,<Instance>,<Data- base>,<User>,<Pass- word>,<Version> Beispiel server,sde:sqlserver:ser- ver\SQLEXPRESS,,test,info- grips,
	SDE Service	Server: <Server> Server mit SDE Service Instance: <SDE Service Port> Port des SDE Services Database: <Database> (Optional) Datenbank des Users. User: <User> Datenbank User Password: <Password> Password des Users	Parameter <Server>,<Instance>,<Data- base>,<User>,<Pass- word>,<Version> Beispiel server,5151,,test,infogrips,

		Version: <Version> (Optional) Leer oder Default dbo.DEFAULT.	
--	--	--	--

2.5. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tablename des OUT Objekts.
OUT.GDB_DATA-SET(s)	o	Geodatabase Dataset, in welches das Objekt geschrieben werden soll.
OUT.<Attribut>(s)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Typ des Werts muss mit dem Typ des Attributs in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden.

Dieses Objektmodell gilt für die Anwendung mit der Prozedur `GDBOUT_WRITE_OBJECT0`. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfiguration mit Prozeduren und Methoden vorgängig erzeugt werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, so beachten Sie das Kapitel mit den Record-Definitionen und das Kapitel mit der Datenbank Modellgenerierung.

Den Objekten wird automatisiert im Attribute `OBJECTID` ein eindeutiger Schlüssel vergeben. Das Attribut `OBJECTID` ist nicht zu definieren.

2.6. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank inklusive dem Geodatabase Repository angelegt werden. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<n>
  TABLE => <Table-Name>
  GDB_DATASET => <Dataset-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix `RECORD_` beginnen und eine eindeutige Nummer `<n>` für die Record Definition beinhalten.

TABLE

Diese Komponente ist required und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente `TABLE` die Datenbank-Tabelle.

GDB_DATASET

Diese Komponente ist optional und definiert das Geodatabase Dataset.



Im GDB-Repository können nur Objekte mit einer Geometrie einem Dataset zugeordnet werden.

Wird diese Komponente bei einem Objekt definiert, das keine Geometrie aufweist, wird eine Dummy-Geometrie in der Form eines Punktes dem Objekt angefügt. Damit kann das Objekt dem Dataset zugeordnet werden.

Um ein Objekt, das keine Geometrie aufweist, als reine Tabelle zu transferieren, darf diese Komponente nicht definiert werden.

<Dataset-Name>

Definiert als Wert der Komponente GDB_DATASET das Geodatabase Dataset..

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

CHAR(<length>)

Stringtypen sind immer als CHAR mit der Länge <length> zu definieren..

NUMBER(p,s)

Number-Type.

INTEGER

Integer-Type.

DATE

Date-Type.

Ein Datum kann übergeben werden als:

- INTEGER im Format YYYYMMDD z.B.20141204

- STRING im Format 'YYYY-MM-DD' z.B '2014-12-04'

Ein Datum mit Zeit kann übergeben werden als

- STRING im Format 'YYYY-MM-DD HH24-MI-SS' z.B '2014-12-04 14:09:59'

GEOMETRY(<type>;<dimension>;<HASM>)

Geometrien müssen als Type GEOMETRY definiert werden. Dabei sind folgende weiteren Definitionen notwendig.

<type>

Type der Geometrie, einer der Werte: point|line|area.

<dimension>

Dimension der Geometrie, einer der Werte: 2D|3D.

<HASM>

Geometrie besitzt die Measure-Dimension, einer der Werte: TRUE|FALSE.



Geodatabase erlaubt nur eine Geometrie-Definition pro Tabelle. Deshalb kann pro Record-Definition nur ein Geometrie-Attribut definiert werden.



Um mögliche Einschränkungen von SDE zu umgehen, ist es empfehlenswert, den Geometrie-Attributen den Name SHAPE zu vergeben.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können absolute Werte oder ICS-Variablen, die einen Wert beinhalten - z.B. IN.OBJID - verwendet werden.

Beispiel einer Record Definition.

```
MAP RECORD_15
  TABLE => Bodenbedeckung_BoFlaeche_Area
  GDB_DATASET => Bodenbedeckung
  OBJID => CHAR(32), IN.OBJID
  Entstehung => CHAR(32), IN.Entstehung.OBJID
  Qualitaet => INTEGER, IN.Qualitaet
  Qualitaet_TXT => CHAR(7), IN.Qualitaet
  Art => INTEGER, IN.Art
  Art_TXT => CHAR(47), IN.Art_TXT
  SHAPE => GEOMETRY(area;2D;FALSE), IN.GEOM
END_MAP
```

2.7. Datasets

Zur Verwaltung von Datasets in der Datenbank ist das analoge Kapitel des Modules DBOUT zu beachten.

Um ein Dataset aus einer Geodatabase Datenbank zu löschen, steht folgende Konfiguration zur Verfügung:

```
ILTOOLS_DIR\system\script\il2arcgis\arcgisdatasetdelete.cfg
```

2.8. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MODEL

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell erstellt werden soll, so ist der Parameter CONFIG_PARAM.GENERATE_MODEL auf ON zu setzen:

```
MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell inklusive den Definitionen für das Geodatabase Repository generiert, falls es nicht schon generiert wurde.

Um das Datenmodell inklusive den Daten und den Definitionen im Geodatabase Repository aus einer Geodatabase Datenbank zu löschen, steht folgende Konfiguration zur Verfügung:

```
ILTOOLS_DIR\system\script\il2arcgis\arcgisgdbdelete.cfg
```

2.9. Topologie Flächennetze

ESRI-Datenbanken - insbesondere SDE - sind sehr heikel bezüglich der Topologie von Flächennetzen. Wenn von INTERLIS Flächennetze wie die Bodenbedeckung der amtlichen Vermessungen mit erlaubten Overlaps übertragen werden, entstehen in der Regel in ESRI-Datenbanken Flächennetze mit Fehlern in der Topologie.

Um solche Fehler in der ESRI-Topologie zu eliminieren, empfehlen wir folgendes Vorgehen.

Konfiguration *.cfg

Definieren Sie die Map ARCGISOUT_SPECIAL

```
MAP ARCGISOUT_SPECIAL
  Bo_BoFlaeche_Area    => AREA_TOPOLOGY
  Li_Liegenschaft_Area => AREA_TOPOLOGY
  DEFAULT              => OFF
END_MAP
```

Erstellen Sie für jede ArcGIS-Featureclass, die ein Flächennetz beinhaltet, einen Eintrag mit dem Wert AREA_TOPOLOGY.

Für diese Featureclasses wird nach der INTERLIS-Topologieberechnung die Topologie ein zweites Mal berechnet, indem Overlap-Flächen als eigenständige Flächen behandelt werden.

Python Script *.py mit Repair Geometry und Validate Topology

Erstellen Sie ein ArcGIS Python Script, das auf Featureclasses mit einem Flächennetz ein Repair Geometry und ein Validate Topology ausführt.

Führen Sie das Python-Script nach dem Datentransfer auf den gewünschten Featureclasses aus oder integrieren Sie das Script in den Datentransfer - siehe nächster Schritt Konfiguration *.out.

Siehe Beispiel Python-Script:

ILTOOLS_DIR\system\script\il2arcgis\il2arcgis_valTopo.py

```
# infoGrips
# 2016.05.31/tg created
#
#=====
import glob
import os
import getopt
import arcpy
import sys

#=====
def validateTopology(featureClass):

    topologyName = featureClass + '_Topology_Validate'

    try:
        arcpy.RepairGeometry_management(in_features=featureClass)
        print(arcpy.GetMessages())
    except:
        print(arcpy.GetMessages())
        raise
    try:
        '''
        If the Cluster Tolerance parameter is blank or set to 0
        the xy tolerance of the feature dataset which contains the topology will be used.
        '''
        arcpy.CreateTopology_management(arcpy.env.workspace, topologyName, 0)
        print(arcpy.GetMessages())
    except:
        print(arcpy.GetMessages())
        raise
    try:
```

```

        arcpy.AddFeatureClassToTopology_management(arcpy.env.workspace + os.sep +
                                                    topologyName, featureClass, "1", "1")

    print(arcpy.GetMessages())
except:
    print(arcpy.GetMessages())
    raise
try:
    arcpy.AddRuleToTopology_management(arcpy.env.workspace + os.sep + topologyName,
                                        "Must Not Have Gaps (Area)", featureClass, "", "", "")

    print(arcpy.GetMessages())
except:
    print(arcpy.GetMessages())
    raise
try:
    arcpy.AddRuleToTopology_management(arcpy.env.workspace + os.sep + topologyName,
                                        "Must Not Overlap (Area)", featureClass, "", "", "")

    print(arcpy.GetMessages())
except:
    print(arcpy.GetMessages())
    raise
try:
    arcpy.ValidateTopology_management(arcpy.env.workspace + os.sep + topologyName,
                                     "FULL_EXTENT")

    print(arcpy.GetMessages())
except:
    print(arcpy.GetMessages())
    return

#=====
def main(argv):

    gdb = ''
    dataset = ''
    feature = ''

    use = 'topo.py -g <gdb> -d <dataset> -f <feature>'

    try:
        opts, args = getopt.getopt(argv,"hg:d:f:")
    except getopt.GetoptError:
        print use
        sys.exit(2)
    for opt, arg in opts:
        if opt == '-h':
            print use
            sys.exit()
        elif opt in ("-g"):
            gdb = arg
        elif opt in ("-d"):
            dataset = arg
        elif opt in ("-f"):
            feature = arg

    print '-g = ', gdb
    print '-d = ', dataset
    print '-f = ', feature

    arcpy.env.overwriteOutput=True

```

```

for gdbFile in glob.glob(gdb):
    arcpy.env.workspace = gdbFile + os.sep + dataset
    print(gdbFile)
    print(arcpy.env.workspace)
    validateTopology(featureClass= feature)
    pass

=====
if __name__ == '__main__':
    main(sys.argv[1:])
=====

```

Konfiguration *.out

Das oben aufgeführte Python-Script kann in den Datentransfer integriert werden.

Siehe Beispiel:

ILTOOLS_DIR\system\script\il2arcgis\il2arcgis.out

```

PROCEDURE ARCGIS_PYTHON_VALTOPO_RUN ! [s dataset, s feature]

! arguments
!-----
=> LOCAL.FEATURE
=> LOCAL.DATASET

! edit
!-----
'C:\Python27\ArcGIS10.2\python.exe' => LOCAL.PY_EXE

'\script\il2arcgis\il2arcgis_valTopo.py' EXPAND_PATH => LOCAL.PY_FILE

! message
!-----
DISPLAY ''
DISPLAY '===== '
DISPLAY 'Topology postprocess'
DISPLAY LOCAL.DATASET, '.', LOCAL.FEATURE
DISPLAY ''

! py and log get
!-----

LOCAL.PY_FILE
FILENAME_PARSE
=> LOCAL.FDIR
=> LOCAL.FNAME
=> LOCAL.FEXT

LOCAL.FDIR . '\' . LOCAL.FNAME . '.log' => LOCAL.LOG

! command build and run
!-----
LOCAL.PY_FILE
. ' -g ' . OPT.output
. ' -d ' . LOCAL.DATASET

```

```

. ' -f ' . LOCAL.FEATURE
=> LOCAL.PY_COMMAND

LOCAL.PY_EXE
. ' ' . LOCAL.PY_COMMAND
. ' > ' . LOCAL.LOG
=> LOCAL.COMMAND

DISPLAY LOCAL.COMMAND
DISPLAY ''

LOCAL.COMMAND OSCALL TO_INT => LOCAL.STATUS

! log add
!-----
IF LOCAL.LOG TEXTFILE.OPEN THEN
  => LOCAL.FID
  WHILE LOCAL.FID TEXTFILE.READLN DO
    DISP
  END_WHILE
END_IF

! message end
!-----
DISPLAY ''
IF LOCAL.STATUS = 0 THEN
  DISPLAY 'finished with success status=',LOCAL.STATUS
ELSE
  DISPLAY 'finished with failure status=',LOCAL.STATUS
END_IF
DISPLAY ''

END_PROCEDURE

!-----

PROCEDURE POST_RUN1

  'Bodenbedeckung' 'Bo_BoFlaeche_Area'  ARCGIS_PYTHON_VALTOPO_RUN
  'Liegenschaften' 'Li_Liegenschaft_Area' ARCGIS_PYTHON_VALTOPO_RUN

END_PROCEDURE

```

2.10. Exportierte Prozeduren und Methoden

Prozedur	ARCGISOUT_OPEN [s input][[]
Beschreibung	<p>Öffnet eine bestehende Datenbank oder kreiert neue Datenbank.</p> <ul style="list-style-type: none"> • Für eine Personal Geodatabase muss in input die Access-Datenbank *.mdb gesetzt werden. • Für eine File Geodatabase muss in input die File-Datenbank *.gdb gesetzt werden.

- Für eine SDE Geodatabase muss in input die SDE connection in der Form <server>, <instance>, <database>, <user>, <password>, <version> gesetzt werden.

Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
'c:\data\arcgis.mdb' ARCGISOUT_OPEN
```

Prozedur

ARCGISOUT_WRITE_OBJECT0

Beschreibung

Schreibt ein Objekt in die Datenbank. Das OUT-Objekt muss gemäss dem Objektmodell gefüllt sein.

Beispiel

```
ARCGISOUT_WRITE_OBJECT0
```

Prozedur

ARCGISOUT_WRITE_RECORD1 ! s recordname

Beschreibung

Schreibt ein Objekt definiert mit <recordname> in die Datenbank. <recordname> ist der Name einer Record-Definition (siehe weiter oben).

Beispiel

```
... => ARCGISOUT_WRITE_RECORD1,RECORD_1
```

Prozedur

ARCGISOUT_CLOSE [[]]

Beschreibung

Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
ARCGISOUT_CLOSE
```

Methode

ARCGISOUT.COMPRESS [[]]

Beschreibung

Eine SDE Datenbank komprimieren. Danach sollten die Anzahl SDE-States minimiert sein: `select count(*) from sde.states`. Die Datenbank muss geöffnet und geschlossen werden. Diese Methode ist für die alleinige Anwendung ohne Import oder Export gedacht. Soll nach dem Import eine Komprimierung erfolgen, dann ist folgender Parameter zu setzen: `ARCGISOUT_PARAM.COMPRESS`

Beispiel

```
<sde-CONNECT-parameter> ARCGIS.OPEN
ARCGIS.COMPRESS
ARCGIS.CLOSE
```

Neben diesen Prozeduren des Modules stehen auch die Methoden der Klasse DB zur Verfügung (s.a. iG/Script Benutzer- und Referenzhandbuch).

2.11. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,
! erzeugt eine Tabelle für LFP's in der Datenbank
! inklusive den Definitionen des Geodatabase Repository
! und schreibt die LFP's in die Tabelle.
```

```
|LICENSE \license\iltoolspro.lic
```

```
MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
```

```
MAP USER_INPUT1
```

```
DIALOG      => FILE
MESSAGE     => 'Enter INTERLIS Input File'
FILE_FILTER => itf
FILE_EXISTS => TRUE
OPT         => input
END_MAP

MAP USER_INPUT2
DIALOG      => ARCGIS
MESSAGE     => 'Enter SDE or Database Output File'
FILE_FILTER => *
FILE_EXISTS => FALSE
OPT         => output
END_MAP

MAP ILIN_PARAM
INTERLIS_DEF => \models\Grunddatensatz.ili
STATISTICS  => ON
CALC_SURFACE => ON
ENUM_TO_TEXT => ON
TRACE       => OFF
END_MAP

MAP ILIN_TOPO
DEFAULT    => OFF
END_MAP

MAP ARCGISOUT_PARAM
SRID       => 'CH1903 LV03'
STATISTICS => ON
DATASET    => ON
END_MAP

MAP RECORD_1
TABLE      => Fi_LFP1
GDB_DATASET => FixpunkteKategoriel
OBJID      => CHAR(32),IN.OBJID
Entstehung => CHAR(32),IN.Entstehung.OBJID
NBIdent    => CHAR(12),IN.NBIdent
Nummer     => CHAR(12),IN.Nummer
SHAPE      => GEOMETRY(point;2D;FALSE),IN.Geometrie
HoeheGeom  => NUMBER(7,3),IN.HoeheGeom
LageGen    => NUMBER(4,1),IN.LageGen
LageZuv    => INTEGER,IN.LageZuv
LageZuv_TXT => CHAR(4),IN.LageZuv_TXT
HoeheGen   => NUMBER(4,1),IN.HoeheGen
HoeheZuv   => INTEGER,IN.HoeheZuv
HoeheZuv_TXT => CHAR(4),IN.HoeheZuv_TXT
Begehbarkeit => INTEGER,IN.Begehbarkeit
Begehbarkeit_TXT => CHAR(14),IN.Begehbarkeit_TXT
Punktzeichen => INTEGER,IN.Punktzeichen
Punktzeichen_TXT => CHAR(17),IN.Punktzeichen_TXT
END_MAP

MAP INPUT_SOURCES
I1 => ILTOPO,OPT.input
END_MAP
```

```

MAP INOUT
  I1                      => IN.TOPIC, IN.TABLE
  I1, Fixpunkte, LFP     => ARCGISOUT_WRITE_RECORD1, RECORD_1
  I1, *                  => OFF
END_MAP

| INCL \script\iltopo.mod
| INCL \script\arcgisout.mod
| INCL \script\il2arcgis\il2arcgis\arcgisout.mod
| INCL \script\run1.prg

```

2.12. Bestehende Konfigurationen IL2GDB/IL2SDE oder GDB2IL/SDE2IL nach IL2ARCGIS/ARCGIS2IL migrieren

Die Module ARCGISIN/ARCGISOUT lösen die Module GDBIN/GDBOUT und SDEIN/SDEOUT ab. Bestehende Konfigurationen IL2GDB/GDB2IL und IL2SDE/SDE2IL sind deshalb durch Konfiguration IL2ARCGIS/ARCGIS2il abzulösen. Für dieser Migration steht folgendes Script zur Verfügung.

```

ILTTOOLS\system\script\il2gdb\CFG_GDB2ARCGIS.cfg

```

Das Script verlangt als Input eine IL2GDB/GDB2IL oder IL2SDE/SDE2IL Konfiguration und schreibt als Output eine analoge IL2ARCGIS/ARCGIS2IL Konfiguration.

3. Modul DBOUT - ODBC-Datenbank schreiben

3.1. Allgemeines

Mit dem Skriptmodul können Objekte in eine ODBC-Datenbank (z.B. MS-Access, Oracle) geschrieben werden.

Der Modul wird mit:

```

| INCL \script\dbout.mod

```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

3.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

3.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.

PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

3.4. Parametermap DBOUT_PARAM

Folgende Parameter können in der Map DBOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
CREATE_TABLE	o	ON, OFF, Default = OFF. Definiert ob die Tables definiert mit Record-Definitionen (siehe weiter unten) in der Datenbank erzeugt werden sollen. Mit OFF werden die Tabellen nicht erzeugt. Mit ON werden die Tabellen erzeugt und zwar zum Zeitpunkt, wenn das erste Objekt in die Tabelle geschrieben wird. Tables in die keine Objekte geschrieben werden, werden auch nicht erzeugt.
DELETE_OLD	o	ON, OFF oder DROP, Default = OFF. Löscht bestehende Daten in Tables definiert mit Record-Definitionen (siehe weiter unten). Mit OFF werden keine Daten gelöscht. Mit ON werden die Daten über ein delete-sql-statement gelöscht. Mit DROP wird die Table und damit die Daten gelöscht. Zusammen mit DROP und dem Parameter CREATE_TABLE => ON werden die Tables gelöscht und wieder erzeugt.
BATCH	o	ON oder OFF, Default = OFF. Mit OFF werden die sql-statements direkt auf der Datenbank ausgeführt. Mit ON werden die sql-statements in ein Batchfile geschrieben. Mit dem Parameter BATCH_FILE wird das Batchfile definiert.
BATCH_OUTPUT_DIR	o	<directory>. Definiert ein Output-Directory für Batchfiles. Mit diesem Parameter kann das Output-Directory für Batchfiles definiert werden, falls der Parameter BATCH = ON definiert ist. Batchfiles können sein ein File mit SQL-Statementes oder in Kombination mit dem Oracle Output Modul die SQLLOADER-Bulkfiles. Ist dieser Parameter nicht gesetzt, so wird das Output-Directory aus einem eventuellen Input-File definiert in OPT.input bestimmt. Ist kein Input-File definiert, so ist das Output-Directory iltools\data\ics.sql.
BATCH_FILE	o	<file>. Definiert das Batchfile. Mit diesem Parameter kann das Batchfile definiert werden, falls der Parameter BATCH = ON definiert ist. Ist dieser Parameter nicht gesetzt, so wird das Batchfile aus einem eventuellen Input-File definiert in OPT.input mit der Endung .sql bestimmt. Ist kein Input-File definiert, so ist das Batchfile iltools\data\ics.sql als definiert. Das Batchfile beinhaltet SQL-Statements, um die transferierten Daten mittels SQL in eine Datenbank zu importieren.
SQLTRACE	o	ON oder OFF, Default = OFF. Zeigt als sql-statements im Logfile an.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltet werden. Sie dazu mehr unter dem Kapitel Datasets. Der Wert ON hebt den Parameter DELETE_OLD auf.

3.5. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tablename des OUT Objekts.
OUT.<Attribut>(s)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Typ des Werts muss mit dem Typ des Attributs in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden.

Dieses Objektmodell gilt für die Anwendung mit der Prozedur `DBOUT_WRITE_OBJECT0`. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfiguration mit Prozeduren und Methoden vorgängig erzeugt werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, so beachten Sie das Kapitel mit den Record-Definitionen und die Prozedur `DBOUT_WRITE_RECORD1`.

3.6. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank angelegt werden. Die Record Definitionen werden von der Prozedur `DBOUT_WRITE_RECORD1` verarbeitet. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<n>
  TABLE => <Table-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix `RECORD_` beginnen und eine eindeutige Nummer `<n>` für die Record Definition beinhalten.

TABLE

Diese Komponente ist required und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente `TABLE` die Datenbank-Tabelle.

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

Alle Datenbanken

CHAR(<length>)

Stringtypen sind immer als CHAR mit der Länge <length> zu definieren. Je nach Datenbank wird der Typ vom Modul in einen der Datenbank bekannten Type umgewandelt z.B. Oracle als VARCHAR2.

NUMBER

Number-Typen ohne Argumente werden vom Modul als NUMBER(38,5) interpretiert.

DB_GEOMETRY(<type>;<dimension>)

Geometrien müssen als Type DB_GEOMETRY definiert werden. Dabei sind folgende weiteren Definitionen notwendig.

<type>

Type der Geometrie, einer der Werte: point|line|area.

<dimension>

Dimension der Geometrie, einer der Werte: 2D|3D.

Der Type **point** wird in 2 Attribute (2D) oder 3 Attribute (3D) mit den Suffixen X,Y,Z zum Attributnamen abgelegt.

Record-Definition:

```
MAP RECORD_1
  TABLE => Bodenbedeckung_BoFlaeche_Area
  Geometrie => DB_GEOMETRY(point;2D),IN.Geometrie
END_MAP
```

Generiert Table:

```
CREATE TABLE Bodenbedeckung_BoFlaeche_Area (
  Geometrie_X DOUBLE, -- X-Koordinate
  Geometrie_Y DOUBLE, -- Y-Koordinate
);
```

Die Type **line** wird in einer Zusatztabelle abgelegt. Der Name der Zusatztabelle ergibt sich aus der Haupttabelle und der Suffix des Geometrieattributes.

Record-Definition:

```
MAP RECORD_1
  TABLE => Bodenbedeckung_BoFlaeche_Geometrie
  GEOM => DB_GEOMETRY(line;2D),IN.GEOM
END_MAP
```

Generiert Table:

```
CREATE TABLE Bodenbedeckung_BoFlaeche_Geometrie (
  GEOM INTEGER
);

CREATE TABLE Bodenbedeckung_BoFlaeche_Geometrie_GEOM (
  ID          INTEGER, -- Fremdschlüssel zu Haupttabelle.GEOM
  POINT_I     INTEGER, -- Punktindex für Punktreihenfolge
  ARCPOINT    INTEGER, -- Ist Punkt ein Punkt auf einem Kreisbogen (1) oder nicht (0)
  X           DOUBLE,  -- X-Koordinate
  Y           DOUBLE,  -- Y-Koordinate
);
```

Der Type **area** wird in einer Zusatztabelle abgelegt. Der Name der Zusatztabelle ergibt sich aus der Haupttabelle und der Suffix des Geometrieattributes.

Record-Definition:

```
MAP RECORD_1
  TABLE => Bodenbedeckung_BoFlaeche_Area
  GEOM => DB_GEOOMETRY(area;2D), IN.GEOM
END_MAP
```

Generiert Table:

```
CREATE TABLE Bodenbedeckung_BoFlaeche_Area (
  GEOM INTEGER
);

CREATE TABLE Bodenbedeckung_BoFlaeche_Area_GEOM (
  ID          INTEGER,    -- Fremdschlüssel zu Haupttabelle.GEOM
  RAND_I     INTEGER,    -- Randindex für Randreihenfolge, 1. Rand = Aussenrand, weiter
  POINT_I    INTEGER,    -- Punktindex für Punktreihenfolge innerhalb eines Randes
  ARCPOINT   INTEGER,    -- Ist Punkt ein Punkt auf einem Kreisbogen (1) oder nicht (0)
  X          DOUBLE,     -- X-Koordinate
  Y          DOUBLE,     -- Y-Koordinate
);
```

MSACCESS**MEMO**

Stringtyp für Texte > 255 Zeichen.

DATETIME

Datums/Zeit Typ. Der <Attribute-Value> muss der SQL-Spezifikation von MSACCESS entsprechen. Zum Beispiel für ein Datum <Attribute-Value> = '03.04.1993', für Datum/Zeit <Attribute-Value>='03.04.1993 17:34:00'.

ORACLE**DATE("YYYY-MM-DD")**

Datums Typ. Mit dem Datums Typ muss auch das Format des Datums definiert werden. Der Modul erzeugt aufgrund des Formats eine entsprechende SQL-Anweisung TO_DATE('1993-04-03', 'YYYY-MM-DD'). Der Wert für das Attribut, muss als Integer oder String die Form YYYYMMDD aufweisen. Zum Beispiel <Attribute-Value>=19939493 oder <Attribute-Value>='19939493'.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können absolute Werte oder ICS-Variablen, die einen Wert beinhalten - z.B. IN.OBJID - verwendet werden.

Beispiel einer Record Definition.

```
MAP RECORD_1
  TABLE => Bodenbedeckung_BoFlaeche_Area
  OBJID => CHAR(10), IN.OBJID
  Entstehung => CHAR(10), IN.Entstehung.OBJID
  Geometrie => DB_GEOOMETRY(point;2D), IN.Geometrie
  Qualitaet => CHAR(30), IN.Qualitaet
  Art => INTEGER, IN.Art
  Art_TXT => CHAR(47), IN.Art_TXT
  Herkunft => CHAR(30), IN.Herkunft
  GEOM => DB_GEOOMETRY(area;2D), IN.GEOM
END_MAP
```

3.7. Datasets

Mit dem Parameter `DBOUT_PARAM.DATASET => ON` kann der Modul veranlasst werden, die Daten in Datasets zu verwalten. In der Regel werden verschiedene Datasets in einer Datenbank geschrieben. Die einzelnen Datasets können danach als Subset des Daten nachgeführt oder gelöscht werden. Typischerweise bildet ein INTERLIS-File ein solches Dataset. Falls der Dataset-Parameter eingeschaltet ist, wird in der Datenbank folgendes angelegt:

```
CREATE TABLE GS_FILE (
  FileID      INTEGER    -- Schlüssel für Dataset
  ModelName   CHAR(255)  -- Modellname (INTERLIS)
  FileName    CHAR(255)  -- Filenamen
  DateFile    INTEGER    -- YYYYMMDD Datum des Files
  DateUpload  INTEGER    -- YYYYMMDD Datum des lesen in die Datenbank
  UserUpload  CHAR(255)  -- eventuell ein Username
);

CREATE TABLE <Data-Table> (
  GS_FileID   INTEGER    -- Fremdschlüssel für Dataset
  :
);
```

Die Tabelle `GS_FILE` verwaltet die Datasets. Jedes Dataset erhält einen eindeutigen Schlüssel im Attribut `FileID`. Jede angelegte Tabelle für die Daten erhält ein Attribut `GS_FileID`, das den Fremdschlüssel des Datasets beinhaltet.

Wird ein Dataset (File) das erste Mal in die Datenbank geschrieben (`INSERT`), wird der Eintrag in `GS_FILE` generiert, und jeder Daten-Record erhält den Fremdschlüssel des Datasets.

Wird ein Dataset (File) ein nächstes Mal in die Datenbank geschrieben (`UPDATE`), wird der Eintrag in `GS_FILE` mit den Daten nachgeführt, die bestehenden Daten des Datasets in der Datenbank gelöscht und die neuen Daten in die Datenbank geschrieben.

Soll ein Dataset (File) aus der Datenbank gelöscht werden (`DELETE`), so steht folgende Konfiguration zur Verfügung:

```
ILTOOLS_DIR\system\script\il2db\dbdatasetdelete.cfg
```

Diese Konfiguration löscht die Daten eines Datasets und den Eintrag des Datasets in `GS_FILE`.

3.8. Datenbank Modellgenerierung mit `CONFIG_PARAM.GENERATE_MODEL`

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell erstellt werden soll, so ist der Parameter `CONFIG_PARAM.GENERATE_MODEL` auf `ON` zu setzen und das Script `il2db.lib` zu includen:

```
MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
:
|INCL \script\il2db\il2db.lib
```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell angelegt, falls es nicht schon angelegt wurde.

3.9. Exportierte Prozeduren und Methoden

Prozedur	<code>DBOUT_OPEN [] []</code>
Beschreibung	Öffnet eine Datenbank definiert mit <code>DB_PARAM.SOURCE</code> . Die Prozedur wird von <code>RUN1</code> automatisch aufgerufen.
Beispiel	<code>DBOUT_OPEN</code>
Prozedur	<code>DBOUT_WRITE_OBJECT0</code>
Beschreibung	Schreibt ein Objekt in die Datenbank. Das <code>OUT</code> -Objekt muss gemäss dem Objektmodell gefüllt sein.
Beispiel	<code>DBOUT_WRITE_OBJECT0</code>
Prozedur	<code>DBOUT_WRITE_RECORD1 ! s recordname</code>
Beschreibung	Schreibt ein Objekt definiert in <code><recordname></code> in die Datenbank. <code><recordname></code> ist der Name einer Record-Definition (siehe weiter oben). Je nach den gesetzten Parametern führt die Prozedur beim ersten Aufruf für eine Record-Definition folgende Aktionen auf der Datenbank aus: <ol style="list-style-type: none"> 1. Die Tabelle oder deren Inhalt wird gelöscht, falls definiert durch Parameter. 2. Die Tabelle wird erzeugt, falls definiert durch Parameter. 3. Das Objekt wird in die Datenbank geschrieben.
Beispiel	<code>... => DBOUT_WRITE_RECORD1,RECORD_1</code>
Prozedur	<code>DBOUT_CLOSE [] []</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von <code>RUN1</code> automatisch aufgerufen.
Beispiel	<code>DBOUT_CLOSE</code>

Neben diesen Prozeduren des Modules stehen auch die Methoden der Klasse `DB` zur Verfügung (s.a. iG/Script Benutzer- und Referenzhandbuch).

3.10. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,
! erzeugt eine Tabelle für LFP's in der Datenbank
! und schreibt die LFP's in die Tabelle.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter INTERLIS Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT         => input
END_MAP
```

```
MAP USER_INPUT2
  DIALOG      => ODBC
  OPT         => output
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\Grunddatensatz.ili
  STATISTICS   => ON
  CALC_SURFACE => ON
  ENUM_TO_TEXT => ON
  TRACE        => OFF
END_MAP

MAP ILIN_TOPO
  DEFAULT => OFF
END_MAP

MAP DB_PARAM
  SOURCE      => '' ! ODBC-Source
  USER        => '' ! ODBC-User
  PASSWD      => '' ! ODBC-Password
  TRACE       => OFF
END_MAP

MAP DBOUT_PARAM
  STATISTICS   => ON
  CREATE_TABLE => ON
  DATASET      => ON
END_MAP

MAP RECORD_1
  TABLE => Fixpunkte_LFP
  OBJID  => CHAR(10),IN.OBJID
  ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
  NUMMER  => CHAR(12),IN.Nummer
  GEOMETRIE => DB_GEOMETRY(point;3D),IN.Geometrie
  LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
  HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
  BEGEHBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
  SYMBOLORI => NUMBER,IN.SymbolOri
  ART_TXT => CHAR(4),IN.Art_TXT
  HERKUNFT => CHAR(30),IN.Herkunft
END_MAP

MAP INPUT_SOURCES
  I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
  I1          => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => DBOUT_WRITE_RECORD1,RECORD_1
  I1,*       => OFF
END_MAP

|INCL \script\iltopo.mod
|INCL \script\dbout.mod
|INCL \script\run1.prg
```

4. Modul DGNOUT - Bentley Microstation DGN schreiben

4.1. Allgemeines

Mit dem Modul können ICS Objekte direkt in Microstation Designfiles geschrieben werden.



Das analoge Modul MSOUT benötigt zum Schreiben von Designfiles das Produkt Microstation.

Der Modul wird mit:

```
| INCL \script\dgnout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

4.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

4.3. Parametermap DGNOUT_PARAM

Folgende Parameter können in der Map DGNOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
CELL_FILE	o	Zellbibliothek die verwendet wird. Obligatorisch wenn Zellen nach Microstation geschrieben werden.
CELL_SYMOLOGY	o	ON oder OFF. Default OFF. Definiert ,ob beim Schreiben von Zellen die Symboley der Zelle in der Zellbibliothek verwendet werden soll.
RSC_DIR	o	STRING. Definiert den Pfad mit den Microstation-Resource-Files, wie zum Beispiel die Fonts. Wird benötigt um zusätzliche Informationen zu den Objekten verarbeiten zu können. Beispiel Fontname bei Texten.
DEBUG	o	ON oder OFF. Default OFF. Debugmodus ein oder aus.
STATISTICS	o	ON oder OFF. Default OFF. Statistik anzeigen ein oder aus.

4.4. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Je nach OUT-Type werden folgende zusätzlichen Komponenten verlangt.

Allgemeine Komponenten für jedes OUT-Objekt

Komponente	req/opt	Beschreibung
OUT.TYPE(s)	r	Typ des Objekts.
OUT.LEVEL(i)	r	Microstation Level-Nummer des Objekts.
OUT.COLOR(i)	r	Microstation Color des Objekts.
OUT.WEIGHT(i)	r	Microstation Weight des Objekts.

OUT.STYLE(i s)	r	Microstation Style-Nummer oder -Name des Objekts.
OUT.GGROUP(i)	o	Microstation Graphische-Gruppe-Nummer des Objekts. Optional.
OUT.MODEL(s)	o	Microstation Model des Objekts. Optional.
OUT.PRIORITY(i)	o	Microstation Priorität des Objekts. Optional.

Zusätzliche Komponenten für OUT.TYPE = 'POINT'

Wird als Microstation Type LINE (3) geschrieben.

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Punkt-Geometrie des Objekts.

Zusätzliche Komponenten für OUT.TYPE = 'LINE'

Wird als Microstation Type LINE (3), LINESTRING (4), ARC (6) oder COMPLEX_LINESTRING (12) geschrieben. Hängt ab von der Anzahl Punkte und den Subtypes (Lines/Arcs) .

Komponente	req/opt	Beschreibung
OUT.GEOM(l)	r	Linie-Geometrie des Objekts.

Zusätzliche Komponenten für OUT.TYPE = 'AREA'

Wird als Microstation Type SHAPE (6) oder COMPLEX_SHAPE (14) geschrieben. Hängt ab von der Anzahl Punkte und den Subtypes (Lines/Arcs) .

Komponente	req/opt	Beschreibung
OUT.GEOM(a)	r	Flächen-Geometrie des Objekts.

Zusätzliche Komponenten für OUT.TYPE = 'FILLED_AREA'

Wird als Microstation Type SHAPE (6) oder COMPLEX_SHAPE (14) geschrieben. Hängt ab von der Anzahl Punkte und den Subtypes (Lines/Arcs) .

Komponente	req/opt	Beschreibung
OUT.GEOM(a)	r	Flächen-Geometrie des Objekts.
OUT.FILLCOLOR(i)	r	Füllfarbe des Objekts.

Zusätzliche Komponenten für OUT.TYPE = 'TEXT'

Wird als Microstation Type TEXT(17) geschrieben.

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Punkt-Geometrie des Objekts.
OUT.TXT(s)	r	Text des Objekts. Bei Symbolen in der Regel 1 Character.
OUT.TW(r)	r	Textbreite des Objekts. (Microstation: tw=).
OUT.TH(r)	r	Texthöhe des Objekts. (Microstation: th=).
OUT.ROT(r)	r	Rotation des Objekts.
OUT.FONT(i)	r	Font des Objekts. (Microstation: ft=). Bei TEXT ein Textfont. Bei SYMBOL ein Symbolfont.
OUT.JUST(s)	r	Textjustierung des Objekts. (Horizontal: L: Left, C:Center, R:Right; Vertikal: T:Top, C:Center, B:Bottom; Beispiel LC).

Zusätzliche Komponenten für OUT.TYPE = 'TEXT_NODE'

Wird als Microstation Type TEXT_NODE(7) geschrieben.

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Punkt-Geometrie des Objekts.
OUT.TXT(s)	r	Text 1. Zeile des Objekts.
OUT.TXT2(s)	o	Text 2. Zeile des Objekts.
OUT.TXT<n>(s)	o	Text n. Zeile des Objekts.
OUT.TW(r)	r	Textbreite des Objekts. (Microstation: tw=).
OUT.TH(r)	r	Texthöhe des Objekts. (Microstation: th=).
OUT.ROT(r)	r	Rotation des Objekts.
OUT.FONT(i)	r	Font des Objekts. (Microstation: ft=).
OUT.JUST(s)	r	Textjustierung des Objekts. (Horizontal: L: Left, C:Center, R:Right; Vertikal: T:Top, C:Center, B:Bottom; Beispiel LC).
OUT.LINESPACING(r)	r	Linespacing des Objekts. (Microstation: ls=).

Zusätzliche Komponenten für OUT.TYPE = 'CELL'

Wird als Microstation Type CELL (2) geschrieben. Falls es sich bei der Zelle um eine Point-Zelle handelt, werden die OUT-Komponenten LEVEL, COLOR, WEIGHT, STYLE berücksichtigt. Bei einer Graphic-Zelle werden diese Komponenten nicht berücksichtigt.

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Punkt-Geometrie des Objekts.
OUT.CELL(s)	r	Zellnamen des Objektes. Die Zellbibliothek mit der Zelldefinition muss im Microstation-Designfile attached sein.
OUT.ROT(r)	r	Rotation des Objekts.
OUT.SCALE(r)	r	Skalierung des Objektes.
OUT.FILLCELL(b)	o	TRUE FALSE. Wenn TRUE werden gefüllte Flächen einer Zelledefinition gefüllt übertragen. Wenn FALSE werden gefüllten Flächen einer Zelledefinition nicht gefüllt übertragen.
OUT.CELL_BACKGROUND_COLOR_KEEP(b)	o	TRUE FALSE. Wenn TRUE wird die Color von Elementen einer Zelledefinition nicht verändert, wenn die Color der Backgroundcolor entspricht. Wenn FALSE wird die Color von Elementen einer Zelledefinition geändert.

 Die Zelle muss in der attached Celllibrary existieren.

Zusätzliche Komponenten für OUT.TYPE = 'SHARED_CELL'

Wird als Microstation Type SHARED_CELL (35) geschrieben. Falls es sich bei der Zelle um eine Point-Zelle handelt, werden die OUT-Komponenten LEVEL, COLOR, WEIGHT, STYLE berücksichtigt. Bei einer Graphic-Zelle werden diese Komponenten nicht berücksichtigt.

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Punkt-Geometrie des Objekts.
OUT.CELL(s)	r	Zellnamen des Objektes. Die Zelldefinition der Shared Cell muss im Microstationfile definiert sein.
OUT.ROT(r)	r	Rotation des Objekts.

OUT.SCALE(r)	r	Skalierung des Objektes.
--------------	---	--------------------------

 Die Shared Cell Definition zu einer Shared Cell wird wie folgt verarbeitet:

1. Existiert die Shared Cell Definition bereits, wird nichts verändert.
2. Existiert die Shared Cell Definition nicht und ist die Cell in der attached Celllibrary enthalten, wird die Shared Cell Definition aus der Cell in der Celllibrary erzeugt.
3. Existiert die Shared Cell Definition nicht und ist die Cell nicht in der attached Celllibrary enthalten, wird die Shared Cell aus den Geometrien in IN.XGEOM erzeugt, falls vorhanden

4.5. Maps für Signaturen

Den Prozeduren DGNOUT_WRITE_* müssen die Namen von Signaturen übergeben werden. Eine Signatur ist eine Zusammenfassung bestimmter graphischer Eigenschaften (z.B. Level oder Farbe) unter einem Namen. Die Signaturnamen werden in den nachfolgenden Maps der .cfg Datei definiert:

```
MAP LINE_SYMBOLOGY
...
<symbology>    => <style>,<level>,<color>,<weight>
...
END_MAP

MAP TEXT_SYMBOLOGY
...
<symbology>    => <font>,<level>,<color>,<weight>,<tw>,<th>
...
END_MAP

MAP SYMBOL_SYMBOLOGY
...
<symbology>    => <font>,<code>,<level>,<color>,<weight>,<scale>
...
END_MAP

MAP CELL_SYMBOLOGY
...
<symbology>    => <cell>,<scale>
...
END_MAP
```

Die einzelnen Parameter haben folgende Bedeutung:

- <symbology>**
Name der definierten Signatur.
- <style>**
Microstation-Linecode.
- <level>**
Microstation-Level.
- <color>**
Microstation-Color.
- <weight>**
Microstation-Weight.

Microstation-Font.

<tw>

Microstation-Text-Width.

<th>

Microstation-Text-Height.

<code>

Microstation-Symbol-Character als Dezimalwert.

<scale>

Microstation-Skalierung.

<cell>

Microstation-Cellname.

4.6. Map für Textjustierung

Das Modul verfügt über eine Map, die die Abbildung der INTERLIS-Textjustierung - HALIGNMENT, VALIGNMENT zur Microstation-Textjustification beinhaltet. Diese Map sieht wie folgt aus.

```
MAP DGNOUT_ALI_TO_JUST
  0,0 => LT
  0,1 => LT
  0,2 => LC
  0,3 => LB
  0,4 => LB
  1,0 => CT
  1,1 => CT
  1,2 => CC
  1,3 => CB
  1,4 => CB
  2,0 => RT
  2,1 => RT
  2,2 => RC
  2,3 => RB
  2,4 => RB
  DEFAULT => CC
END_MAP
```

Diese Map kann in einer Konfiguration wie folgt angewendet werden.

```
IN.NamHALi , IN.NamVALi DGNOUT_ALI_TO_JUST => OUT.JUST
```

4.7. Exportierte Prozeduren und Methoden

Prozedur DGNOUT_OPEN ! [s output][[]]

Beschreibung Öffnet das Designfile <output> und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
'' DGNOUT_OPEN
```

Prozedur DGNOUT_WRITE_OBJECT ! [[]]

Beschreibung Schreibt ein Objekt in das Designfile. Das OUT-Objekt muss gemäss dem Objektmodell vorbereitet sein.

Beispiel

```
&OUT MAPCLEAR
'POINT'      => OUT.TYPE
1           => OUT.LEVEL
2           => OUT.COLOR
3           => OUT.WEIGHT
4           => OUT.STYLE
IN.Geometrie => OUT.GEOM

DGNOUT_WRITE_OBJECT
```

Prozedur

DGNOUT_CLOSE ! [][]

Beschreibung

Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
DGNOUT_CLOSE
```

Prozedur

DGNOUT_WRITE_POINT2 ! p point, s symbology

Beschreibung

Schreibt einen Punkt in das Designfile. Die Signatur muss in der Map LINE_SYMBLOGY definiert sein.

Beispiel

Definition Symbology.

```
MAP LINE_SYMBLOGY
!SYMBLOGY      => STYLE,LEVEL,COLOR,WEIGHT
SYMB1 => 0,1,1,0
END_MAP
```

Anwendung mit SET_PARAM .

```
IN.Geometrie SET_PARAM
'SYMB1' SET_PARAM
DGNOUT_WRITE_POINT2
```

Anwendung in einer Rule.

```
MAP INOUT
I1 => IN.TOPIC,IN.TABLE
I1,Fixpunkte,LFP => DGNOUT_WRITE_POINT2,IN.Geometrie,SYMB1
END_MAP
```

Prozedur

DGNOUT_WRITE_LINE2 ! l line, s symbology

Beschreibung

Schreibt eine Linie in das Designfile. Die Signatur muss in der Map LINE_SYMBLOGY definiert sein.

Beispiel

Definition Symbology.

```
MAP LINE_SYMBLOGY
!SYMBLOGY      => STYLE,LEVEL,COLOR,WEIGHT
SYMB1 => 0,1,1,0
END_MAP
```

Anwendung mit SET_PARAM .

```
IN.Geometrie SET_PARAM
'SYMB1' SET_PARAM
DGNOUT_WRITE_LINE2
```

Anwendung in einer Rule.

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1, Einzelobjekte, Linienelement => DGNOUT_WRITE_LINESTRING2, ...
                                     ...IN.Geometrie, SYMB1
END_MAP
```

Prozedur

DGNOUT_WRITE_AREA2 ! a area, s symbology

Beschreibung

Schreibt eine Fläche in das Designfile. Die Signatur muss in der Map LINE_SYMBIOLOGY definiert sein.

Beispiel

Definition Symbology.

```
MAP LINE_SYMBIOLOGY
  !SYMBIOLOGY => STYLE, LEVEL, COLOR, WEIGHT
  SYMB1 => 0,1,1,0
END_MAP
```

Anwendung mit SET_PARAM .

```
IN.Geometrie SET_PARAM
'SYMB1' SET_PARAM
DGNOUT_WRITE_AREA2
```

Anwendung in einer Rule.

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1, Einzelobjekte, Flaechenelement => DGNOUT_WRITE_AREA2, ...
                                     ...IN.Geometrie, SYMB1
END_MAP
```

Prozedur

DGNOUT_WRITE_FILLED_AREA2 ! a area, s symbology

Beschreibung

Schreibt eine gefüllte Fläche in das Designfile. Die Signatur muss in der Map LINE_SYMBIOLOGY definiert sein.

Beispiel

Definition Symbology.

```
MAP LINE_SYMBIOLOGY
  !SYMBIOLOGY => STYLE, LEVEL, COLOR, WEIGHT
  SYMB1 => 0,1,1,0
END_MAP
```

Anwendung mit SET_PARAM .

```
IN.Geometrie SET_PARAM
'SYMB1' SET_PARAM
DGNOUT_WRITE_FILLED_AREA2
```

Anwendung in einer Rule.

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1, Einzelobjekte, Flaechenelement => DGNOUT_WRITE_FILLED_AREA2, ...
                                     ...IN.Geometrie, SYMB1
END_MAP
```

Prozedur

DGNOUT_WRITE_TEXT6 ! s text, p position, r rotation, i Hali, i Vali, s symbology

Beschreibung Schreibt einen Text in das Designfile. Die Signatur muss in der Map TEXT_SYMBOLOGY definiert sein. Die Procedure wird kann auch für das Schreiben von Symbolen verwendet.

Beispiel Definition Symbology.

```
MAP TEXT_SYMBOLOGY
  !SYMBOLOGY => FONT,LEVEL,COLOR,WEIGHT,TW,TH
  SYMB1 => 0,1,1,0,1.0,1.0
END_MAP
```

Anwendung mit SET_PARAM .

```
IN.Name SET_PARAM
IN.NamePos SET_PARAM
IN.NameOri SET_PARAM
IN.NameHAlI SET_PARAM
IN.NameVAlI SET_PARAM
'SYMB1' SET_PARAM
DGNOUT_WRITE_TEXT6
```

Anwendung in einer Rule.

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Einzelobjekte,Objektname => DGNOUT_WRITE_TEXT6,...
                                   ...IN.Name,IN.NamPos,...
                                   ...IN.NamOri,IN.NamHAlI,...
                                   ...IN.NamVAlI,SYMB1
END_MAP
```

Prozedur DGNOUT_WRITE_CELL3 ! p position, r rotation, s symbology

Beschreibung Schreibt eine Zellen in das Designfile. Die Signatur muss in der Map CELL_SYMBOLOGY definiert sein.

Beispiel Definition Symbology.

```
MAP CELL_SYMBOLOGY
  !SYMBOLOGY => CELL,SCALE
  SYMB1 => 097041,1.0
END_MAP
```

Anwendung mit SET_PARAM .

```
IN.Geometrie SET_PARAM
0.0 SET_PARAM
'SYMB1' SET_PARAM
DGNOUT_WRITE_CELL3
```

Anwendung in einer Rule.

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => DGNOUT_WRITE_CELL3,...
                                   ...IN.Geometrie,0.0,SYMB1
END_MAP
```

Prozedur DGNOUT_WRITE_SHARED_CELL3 ! p position, r rotation, s symbology

Beschreibung Schreibt eine Zelle in das Designfile. Die Signatur muss in der Map CELL_SYMBOLOGY definiert sein.

Beispiel

Definition Symbology.

```
MAP CELL_SYMBOLOGY
  !SYMBOLOGY      => CELL,SCALE
  SYMB1 => 097041,1.0
END_MAP
```

Anwendung mit SET_PARAM .

```
IN.Geometrie SET_PARAM
0.0 SET_PARAM
'SYMB1' SET_PARAM
DGNOUT_WRITE_SHARED_CELL3
```

Anwendung in einer Rule.

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => DGNOUT_WRITE_SHARED_CELL3,...
                      ...IN.Geometrie,0.0,SYMB1
END_MAP
```

Prozedur**DGNOUT_SET_MODEL1 ! s model****Beschreibung**

Setzt global OUT.MODEL für Objekte, siehe auch Objektmodell

Beispiel

Anwendung mit SET_PARAM .

```
'MyModel' SET_PARAM
DGNOUT_SET_MODEL1
```

Anwendung in einer Rule.

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => DGNOUT_SET_MODEL1,MyModel,...
END_MAP
```

Prozedur**DGNOUT_SET_PRIORITY1 ! i priority****Beschreibung**

Setzt global OUT.PRIORITY für Objekte, siehe auch Objektmodell

Beispiel

Anwendung mit SET_PARAM .

```
100 SET_PARAM
DGNOUT_SET_PRIORITY1
```

Anwendung in einer Rule.

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => DGNOUT_SET_PRIORITY1,100,...
END_MAP
```

Prozedur**DGNOUT_SET_FILLCELL1 ! b TRUE|FALSE****Beschreibung**

Setzt global OUT.FILLCELL für Zellen, siehe auch Objektmodell für OUT.TYP = 'CELL'

Beispiel

Anwendung mit SET_PARAM .

```
TRUE SET_PARAM
DGNOUT_SET_FILLCELL1
```

Anwendung in einer Rule.

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => DGNOUT_SET_FILLCELL1,TRUE,...
END_MAP
```

Prozedur	DGNOUT_SET_CELL_BACKGROUND_COLOR_KEEP1 ! b TRUE FALSE
Beschreibung	Setzt global OUT.CELL_BACKGROUND_COLOR_KEEP1 für Zellen, siehe auch Objektmodell für OUT.TYP = 'CELL'
Beispiel	Anwendung mit SET_PARAM.

```
TRUE SET_PARAM
DGNOUT_SET_CELL_BACKGROUND_COLOR_KEEP1
```

Anwendung in einer Rule.

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => DGNOUT_SET_CELL_BACKGROUND_COLOR_KEEP1,TRUE,...
END_MAP
```

4.8. Skriptbeispiel

Beispiel ohne Datenbankanbindung.

```
! Diese ICS Konfiguration kopiert einige von ilin.mod
! gelesenen Objekte mit dgnout.mod in eine DGN-Datei.
! Eine Datenbankanbindung wird nicht berücksichtigt.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG => FILE
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP USER_INPUT2
  DIALOG => FILE
  MESSAGE => 'Enter .dgn Output File'
  FILE_FILTER => dgn
  FILE_EXISTS => FALSE
  OPT => output
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\Grunddatensatz.ili
  STATISTICS => ON
END_MAP

MAP DGNOUT_PARAM
  STATISTICS => ON
END_MAP
```

```

MAP LINE_SYMBOLOLOGY
  !SYMBOLOLOGY      => STYLE,LEVEL,COLOR,WEIGHT
  SYMB1 => 0,1,1,0
END_MAP

MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
END_MAP

MAP INOUT
  I1                                     => IN.TOPIC,IN.TABLE
  I1,Bodenbedeckung,BoFlaeche_Geometrie => DGNOUT_WRITE_LINE2,IN.GEOM,SYMB1
  I1,Bodenbedeckung,BoFlaeche           => DGNOUT_WRITE_POINT2,IN.Geometrie,SYMB1
  I1,*                                   => OFF
END_MAP

|INCL \script\ilin.mod
|INCL \script\dgnout.mod
|INCL \script\run1.prg

```

5. Modul DXFOUT - AutoCAD DXF schreiben

5.1. Allgemeines

Mit dem Modul DXFOUT können Objekte in eine AutoCAD DXF Datei geschrieben werden.

Der Modul DXFOUT wird mit:

```
|INCL \script\dxfout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

5.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

5.3. Parametermap DXFOUT_PARAM

Folgende Parameter können in der Map DXFOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
TEMPLATE	r	Name der Headerdatei. Für die Headerdatei muss ein DXF-File gemäss Autocad Version 10-14 Standard angegeben werden. Bemerkung: eine allfällig vorhandene ENTITIES Section wird ignoriert.
FILTER_LAYERS	o	Definiert, ob nur die Layer aus der Headerdatei übertragen werden sollen, für die auch Objekte übertragen wurden (ON oder OFF). Per Default werden alle Layer übertragen.
FILTER_BLOCKS	o	Definiert, ob nur die Blockdefinitionen aus der Headerdatei übertragen werden sollen, für die auch Objekte übertragen wurden (ON oder OFF). Per Default werden alle Blockdefinitionen übertragen
DEFAULT_LAYER	o	Name eines Default-Layers. Definiert einen Default-Layer, der einem Element zugeordnet wird, falls der in der Konfiguration

		definierte Layer nicht existiert. Der Default-Layer muss in der Headerdatei existieren.
DEFAULT_BLOCK	o	Name eines Default-Blocks. Definiert einen Default-Block, der einem Element zugeordnet wird, falls der in der Konfiguration definierte Block nicht existiert. Der Default-Block muss in der Headerdatei existieren.
CREATE_LAYERS	o	Definiert ob ein Layer eines übertragenen Elementes im Header erzeugt werden sollen, falls der Layer nicht bereits in der Headerdatei vorhanden ist (ON oder OFF). Per Default werden keine Layer erzeugt.
HANDLES	o	Für jedes Objekt ein DXF Handle erzeugen (ON oder OFF).
DIMENSION	o	2 oder 3. Definiert ob das DXF-File 2D oder 3D Geometrien enthalten soll. Wenn nicht definiert, kann das File 2D und 3D Geometrien enthalten.
UNDEFZ	o	REAL. Definiert eine Z-Koordinate für Geometrien ohne Z-Koordinate.
DEBUG	r	ON oder OFF, Default = OFF. Debugmodus ein oder aus.
STATISTICS	r	ON oder OFF, Default = OFF. Objektstatistik am Ender der .log Datei anzeigen.
PRIORITY	o	ON oder OFF, Default = OFF. Soll PRIORITY verarbeitet werden. Siehe auch Procedure DXFOUT_SET_PRIORITY1.

5.4. Objektmodell

Der Modul verlangt für jedes OUT-Objekt folgende Systemkomponenten:

Pro Objekttyp müssen ausserdem folgende Komponenten des OUT-Objekts gesetzt werden:

Allgemeine Komponenten für jedes IN-Objekt

Komponente	req/opt	Beschreibung
OUT.TYPE(s)	r	Objekttyp (s.a. unten).
OUT.LAYER(s)	r	DXF-Layer.
OUT.LTYPE(s)	o	DXF-Linientyp.
OUT.COLOR(i)	o	DXF-Farbe (0 .. 255).
OUT.THICKNESS(r)	o	DXF-Thickness.

Zusätzliche Komponenten für OUT.TYPE = 'POINT'

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Punktposition.

Zusätzliche Komponenten für OUT.TYPE = 'BLOCK'

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Blockposition.
OUT.BLOCK(s)	r	Blockname.
OUT.SCALE(r)	o	Blockskalierungsfaktor.
OUT.ROT(r)	o	Blockorientierung in Altgrad.

Zusätzliche Komponenten für OUT.TYPE = 'SHAPE'

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Shapeposition.
OUT.SHAPE(s)	r	Shapename.
OUT.SIZE(r)	r	Shapegrösse.
OUT.ROT(r)	r	Shapeorientierung in Altgrad.

Zusätzliche Komponenten für OUT.TYPE = 'CIRCLE'

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Kreiszentrum.
OUT.RADIUS(r)	r	Kreisradius.

Zusätzliche Komponenten für OUT.TYPE = 'TEXT'

Komponente	req/opt	Beschreibung
OUT.TXT(s)	r	Textinhalt.
OUT.GEOM(p)	r	Textposition.
OUT.HEIGHT(r)	r	Texthöhe.
OUT.STYLE(s)	r	Textfont.
OUT.JUST(p)	r	Justierung. Mögliche Werte sind: LEFT, CENTER, RIGHT, MIDDLE.
OUT.SLANT(r)	r	Textneigung in Altgrad.
OUT.XSCALE(r)	r	Textskalierung entlang X-Achse.
OUT.ROT(r)	r	Orientierungswinkel in Altgrad.

Zusätzliche Komponenten für OUT.TYPE = 'LINE'

Komponente	req/opt	Beschreibung
OUT.GEOM(l)	r	Liniengeometrie. Falls die übergebene Linie aus mehreren Linien bzw. Kreisbogenstücken besteht, wird die Liniengeometrie automatisch in Linien und Kreisbögen aufgelöst.

Zusätzliche Komponenten für OUT.TYPE = 'POLYLINE'

Komponente	req/opt	Beschreibung
OUT.GEOM(l, a)	r	Linien- oder Flächengeometrie. Falls eine Flächengeometrie übergeben wird, werden geschlossene Polylines für jeden Rand der Fläche (inkl. Inseln) geschrieben.

Zusätzliche Komponenten für OUT.TYPE = 'BLOCKDEF'

Komponente	req/opt	Beschreibung
OUT.XGEOM(L)	r	Liste von Geometrien für den Block.
OUT.BLOCK(s)	r	Blockname.
OUT.GEOM(p)	o	Punktgeometrie. Wenn definiert werden die Geometrien in OUT.XGEOM um diesen Vektor nach 0.0/0.0 verschoben.
OUT.SCALE(r)	o	Blockskalierungsfaktor. Wenn definiert werden die Geometrien in OUT.XGEOM auf den Skalierungsfaktor 1.0 skaliert.

OUT.ROT(<i>r</i>)	o	Blockorientierung in Altgrad. Wenn definiert werden die Geometrien in OUT.XGEOM auf die Rotation 0.0 rotiert.
---------------------	---	---

5.5. Maps für Signaturen

Den Prozeduren DXFOUT_WRITE_* müssen die Namen von Signaturen übergeben werden. Eine Signatur ist eine Zusammenfassung bestimmter graphischer Eigenschaften (z.B. Layer oder Farbe) unter einem Namen. Die Signaturnamen werden in den nachfolgenden Maps der .cfg Datei definiert:

```
MAP POINT_SYBOLOGY
...
<symbology> => <layer>,<ltype>,<color>,<thickness>
...
END_MAP

MAP BLOCK_SYBOLOGY
...
<symbology> => <layer>,<ltype>,<color>,<thickness>,<block>,<scale>
...
END_MAP

MAP SHAPE_SYBOLOGY
...
<symbology> => <layer>,<ltype>,<color>,<thickness>,<shape>,<size>
...
END_MAP

MAP TEXT_SYBOLOGY
...
<symbology> => <layer>,<ltype>,<color>,<thickness>,<style>,<height>,<xscale>,<slant>
...
END_MAP

MAP LINE_SYBOLOGY
...
<symbology> => <layer>,<ltype>,<color>,<thickness>
...
END_MAP

MAP POLYLINE_SYBOLOGY
...
<symbology> => <layer>,<ltype>,<color>,<thickness>,<width>,<pgroup>,<priority>
...
END_MAP
```

Die einzelnen Parameter haben folgende Bedeutung:

<symbology>

Name der definierten Signatur.

<layer>

DXF-Layer.

<ltype>

Liniensignatur, z.B. DOTTED.

<color>

DXF-Farbwert (0 .. 255).

<thickness>

DXF-Thickness (3D Linienhöhe).

<block>

Blockname.

<scale>

Blockskalierungsfaktor.

<shape>

Shapename.

<size>

Shapeskalierungsfaktor.

<style>

Textfont, z.B. STANDARD.

<height>

Texthöhe in Benutzereinheiten.

<xscale>

Textskalierung in Textrichtung.

<slant>

Textneigungswinkel.

<width>

Linienbreite.

<pgroup>

Prioritätsgruppe. Optional für Polylines. Polylines innerhalb einer Gruppe <pgroup> und mit gleicher Geometrie werden über die <priority> eliminiert.

<priority>

Prioritätswert. Optional für Polylines. Innerhalb einer Gruppe <pgroup> eliminiert eine Polyline mit der <priority> = n eine identische Polyline mit der <priority> <= n.

5.6. DXF Templates

Mit dem Parameter DXFOUT_PARAM.TEMPLATE muss ein DXF-Template definiert werden, das folgende Inhalt aufweisen muss.

```
SECTION HEADER
- Header section mit allen benötigten Definition
- im Format DXF Version 10-14

SECTION TABLES
TABLE LTYPE
- alle verwendeten Linestyles
TABLE LAYER
- alle verwendeten Layers
- mit DXFOUT_PARAM.CREATE_LAYERS => ON können nicht definierte Layers auch erzeugt werden
TABLE STYLE
- alle verwendeten Styles (Fonts)
```

```
SECTION BLOCKS
- alle verwendeten Blocks
```

In der Section Header ist insbesondere darauf zu achten, dass der Wert des Keywords HANDLING auf 0 gesetzt ist.

```
9
$HANDLING
70
0 <--- !!!
```

Ein solches DXF-Template kann wie folgt erstellt werden:

1. Erzeugen Sie mit AutoCAD ein neues DWG-File.
2. Definieren Sie mit AutoCAD alle benötigten Elemente wie Layers und Blocks.
3. Speichern Sie das File mit AutoCAD als DXF Version 14 ab.
4. Editieren Sie das DXF File bei Bedarf (z.B. Anpassung Keyword HANDLING).

5.7. Exportierte Prozeduren und Methoden

Prozedur	<code>DXFOUT_OPEN ! [s file][[]</code>
Beschreibung	Öffnet den DXFOUT Modul auf der Outputdei <file>. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'c:\test.dxf' DXFOUT_OPEN</code>
Prozedur	<code>DXFOUT_WRITE_CIRCLE3 ! p position, r radius, s linesymbology</code>
Beschreibung	Schreibt einen Kreis in die aktuelle Outputdatei.
Beispiel	<code>... => DXFOUT_WRITE_CIRCLE3,IN.Geometrie,3.0,LS1</code>
Prozedur	<code>DXFOUT_WRITE_POINT2 ! p position, s pointsymbology</code>
Beschreibung	Schreibt einen Punkt in die aktuelle Outputdatei.
Beispiel	<code>... => DXFOUT_WRITE_POINT2,IN.Geometrie,PS1</code>
Prozedur	<code>DXFOUT_WRITE_SHAPE3 ! p position, r rotation, s shapelymbology</code>
Beschreibung	Schreibt eine DXF-Shape Symbol in die aktuelle Outputdatei.
Beispiel	<code>... => DXFOUT_WRITE_SHAPE3,IN.Geometrie,30.0,SS1</code>
Prozedur	<code>DXFOUT_WRITE_BLOCK3 ! p position, r rotation, s blocksymbology</code>
Beschreibung	Schreibt eine DXF-Block Symbol in die aktuelle Outputdatei.
Beispiel	<code>... => DXFOUT_WRITE_BLOCK3,IN.Geometrie,30.0,BS1</code>
Prozedur	<code>DXFOUT_WRITE_ATTRIB7 ! s tag, s txt, p pos, r rotation, i hali, i vali, s textsymbology</code>
Beschreibung	Schreibt ein DXF-Attribute in die aktuelle Outputdatei. Ein DXF-Attribute kann nur direkt nach einem DXF-Block geschrieben werden. Das Schreiben eines DXF-Attributes wird nicht im DXF-Konfigurationseditor unterstützt. Damit trotzdem DXF-Attribute mit dem DXF-Konfigurations-

editor geschrieben werden können, muss in die Konfiguration \script\il2dxf.out inkludiert werden. Ein DXF-Attribut wird dann wie folgt geschrieben: Mit der Prozedur ATTRIBTAG1, <Tag> wird der Tag des DXF-Attributes gesetzt. Danach wird mit DXFOUT_WRITE_TEXT6 ein DXF-Text geschrieben. Ist der <Tag> gesetzt, schreibt DXFOUT_WRITE_TEXT6 ein DXF-Attribut anstatt einem DXF-Text.

Beispiel

```
... => DXFOUT_WRITE_ATTRIB7, 'message', 'hello', IN.Geometrie, 30.0, 1, 2, TS1
```

Prozedur

DXFOUT_WRITE_TEXT6 ! s txt, p pos, r rotation, i hali, i vali, s textsymbology

Beschreibung

Schreibt einen DXF-Text in die aktuelle Outputdatei.

Beispiel

```
... => DXFOUT_WRITE_TEXT6, 'hello', IN.Geometrie, 30.0, 1, 2, TS1
```

Prozedur

DXFOUT_WRITE_LINE2 ! l geometry, s linesymbology

Beschreibung

Schreibt einen DXF-Linie in die aktuelle Outputdatei.

Beispiel

```
... => DXFOUT_WRITE_LINE2, IN.GEOM, LS1
```

Prozedur

DXFOUT_WRITE_POLYLINE2 ! l geometry, s polylinesymbology

Beschreibung

Schreibt einen DXF-Linie in die aktuelle Outputdatei.

Beispiel

```
... => DXFOUT_WRITE_LINE2, IN.GEOM, LS1
```

Prozedur

DXFOUT_SET_PRIORITY1 ! i priority

Beschreibung

Es kann eine Priorität gesetzt werden, mit der die nachfolgenden Elemente in das Output-File geschrieben werden. Die Priorität bestimmt die Reihenfolge in der die Elemente in das Output-File geschrieben werden. Ein Element mit einer höheren Priorität wird nach einem Element mit einer tieferen Priorität in das Output-File geschrieben. Ein später geschriebenes Element wird über einem früher geschriebenen Element dargestellt.

Nur wirksam wenn auch DXFOUT_PARAM.PRIORITY => ON gesetzt ist.

Beispiel

```
... => DXFOUT_SET_PRIORITY1, 10, DXFOUT_WRITE_LINE2, IN.GEOM, LS1
... => DXFOUT_SET_PRIORITY1, 20, DXFOUT_WRITE_LINE2, IN.GEOM, LS2
```

Prozedur

DXFOUT_CLOSE ! [[]]

Beschreibung

Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

```
DXFOUT_CLOSE
```

Prozedur

DXFOUT_CREATE_BLOCK ! [[]]

Beschreibung

Spezial Prozedur zur Kreation einer Blocks. Kreiert eine Block-Definition im Outputfile. Für die Prozedur muss das OUT-Objekt gemäss dem Objektmodell für den Typ BLOCKDEF aufbereitet werden.

Beispiel

```
DXFOUT_CREATE_BLOCK
```

5.8. Skriptbeispiel

```
! Diese ICS Konfiguration kopiert alle von DXFIN
! gelesenen Objekte mit DXFOUT in in eine .dxf Datei.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | FILES | DIRECTORY | STRING | ODBC
  MESSAGE => 'Enter .DXF Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP USER_INPUT2
  DIALOG => DIRECTORY ! FILE | FILES | DIRECTORY | STRING | ODBC
  MESSAGE => 'Enter .DXF Output Directory'
  OPT => output
END_MAP

MAP DXFIN_PARAM
  STATISTICS => ON
  DEBUG      => OFF
END_MAP

MAP DXFOUT_PARAM
  STATISTICS => ON
  DEBUG      => OFF
END_MAP

MAP INPUT_SOURCES
  I1 => DXFIN,OPT.input
END_MAP

MAP INOUT
  I1 => COPY_INOUT0
END_MAP

|INCL \script\ilin.mod
|INCL \script\ilout.mod

PROCEDURE COPY_INOUT0
  ! copy IN map
  &IN &OUT MAPCOPY
  DXFOUT_WRITE_OBJECT
END_PROCEDURE

|INCL \script\run1.prg
```

6. Modul FILEGDBOUT - ESRI File-Geodatabase schreiben

6.1. Allgemeines

Mit dem Modul können Objekte in eine ESRI File-Geodatabase geschrieben werden.

Der Modul beinhaltet folgende Besonderheiten:

- Schreibt Daten mit Geometrien in eine Geodatabase.
- Legt Tabellen für die Daten und die Spatial Indexe entsprechende der ESRI File-Geodatabase an.
- Füllt das Geodatabase Repository mit den notwendigen Definitionen.
- Die Geodatabase kann nach dem Schreiben der Daten direkt mit den ESRI-Anwendungen weiterbearbeitet werden.

Der Modul wird mit:

```
| INCL \script\filegdbout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

6.2. ESRI Lizenz

Der Module benötigt **keine** ESRI Lizenz. Der Modul verwendet das ESRI File Geodatabase API.

rcGIS Engine Runtime Lizenz.

6.3. Parametermap FILEGDBOUT_PARAM

Folgende Parameter können in der Map FILEGDBOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SRID	r	<STRING>. Der Name des ArcGIS Spatial Reference System. Es muss der Name eines ArcGIS bekannten Projected Coordinate Systems sein. Beispiele: 'CH1903_LV03', 'CH1903+_LV95'
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltet werden. Sie dazu mehr unter dem Kapitel Datasets im Module DBOUT.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.

6.4. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tablename des OUT Objekts.

OUT.GDB_DATA-SET(s)	o	Geodatabase Dataset, in welches das Objekt geschrieben werden soll.
OUT.<Attribut>(s)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Typ des Werts muss mit dem Typ des Attributs in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden.

Dieses Objektmodell gilt für die Anwendung mit der Prozedur FILEGDBOUT_WRITE_OBJECT0. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfiguration mit Prozeduren und Methoden vorgängig erzeugt werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, so beachten Sie das Kapitel mit den Record-Definitionen und das Kapitel mit der Datenbank Modellgenerierung.

Den Objekten wird automatisiert im Attribute OBJECTID ein eindeutiger Schlüssel vergeben. Das Attribut OBJECTID ist nicht zu definieren.

6.5. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank inklusive dem Geodatabase Repository angelegt werden. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<n>
  TABLE => <Table-Name>
  GDB_DATASET => <Dataset-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix RECORD_ beginnen und eine eindeutige Nummer <n> für die Record Definition beinhalten.

TABLE

Diese Komponente ist required und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente TABLE die Datenbank-Tabelle.

GDB_DATASET

Diese Komponente ist optional und definiert das Geodatabase Dataset.



Im GDB-Repository können nur Objekte mit einer Geometrie einem Dataset zugeordnet werden.

Wird diese Komponente bei einem Objekt definiert, das keine Geometrie aufweist, wird eine Dummy-Geometrie in der Form eines Punktes dem Objekt angefügt. Damit kann das Objekt dem Dataset zugeordnet werden.

Um ein Objekt, das keine Geometrie aufweist, als reine Tabelle zu transferieren, darf diese Komponente nicht definiert werden.

<Dataset-Name>

Definiert als Wert der Komponente GDB_DATASET das Geodatabase Dataset..

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

CHAR(<length>)

Stringtypen sind immer als CHAR mit der Länge <length> zu definieren..

NUMBER(p,s)

Number-Type.

INTEGER

Integer-Type.

DATE

Date-Type.

Ein Datum kann übergeben werden als:

- INTEGER im Format YYYYMMDD z.B.20141204

- STRING im Format 'YYYY-MM-DD' z.B '2014-12-04'

Ein Datum mit Zeit kann übergeben werden als

- STRING im Format 'YYYY-MM-DD HH24-MI-SS' z.B '2014-12-04 14:09:59'

GEOMETRY(<type>;<dimension>;<HASM>)

Geometrien müssen als Type GEOMETRY definiert werden. Dabei sind folgende weiteren Definitionen notwendig.

<type>

Type der Geometrie, einer der Werte: point | line | area.

<dimension>

Dimension der Geometrie, einer der Werte: 2D | 3D.

<HASM>

Geometrie besitzt die Measure-Dimension, einer der Werte: TRUE | FALSE.



Geodatabase erlaubt nur eine Geometrie-Definition pro Tabelle. Deshalb kann pro Record-Definition nur ein Geometrie-Attribut definiert werden.



Um mögliche Einschränkungen von SDE zu umgehen, ist es empfehlenswert, den Geometrie-Attributen den Name SHAPE zu vergeben.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können absolute Werte oder ICS-Variablen, die einen Wert beinhalten - z.B. IN.OBJID - verwendet werden.

Beispiel einer Record Definition.

```
MAP RECORD_15
TABLE => Bodenbedeckung_BoFlaeche_Area
GDB_DATASET => Bodenbedeckung
OBJID => CHAR(32), IN.OBJID
Entstehung => CHAR(32), IN.Entstehung.OBJID
```

```

Qualitaet => INTEGER,IN.Qualitaet
Qualitaet_TXT => CHAR(7),IN.Qualitaet
Art => INTEGER,IN.Art
Art_TXT => CHAR(47),IN.Art_TXT
SHAPE => GEOMETRY(area;2D;FALSE),IN.GEOM
END_MAP

```

6.6. Datasets

Zur Verwaltung von Datasets in der Datenbank ist das analoge Kapitel des Modules DBOUT zu beachten.

Um ein Dataset aus einer Geodatabase Datenbank zu löschen, steht folgende Konfiguration zur Verfügung:

```

ILTOOLS_DIR\system\script\il2filegdb\filegdbdatasetdelete.cfg ! not implemented yet

```

6.7. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MODEL

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell erstellt werden soll, so ist der Parameter CONFIG_PARAM.GENERATE_MODEL auf ON zu setzen:

```

MAP CONFIG_PARAM
GENERATE_MODEL => ON
END_MAP

```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell inklusive den Definitionen für das Geodatabase Repository generiert, falls es nicht schon generiert wurde.

6.8. Exportierte Prozeduren und Methoden

Prozedur	FILEGDBOUT_OPEN [s input][[]]
Beschreibung	Öffnet eine bestehende Datenbank oder kreiert neue Datenbank. Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'c:\data\filegdb.gdb' FILEGDBOUT_OPEN</code>
Prozedur	FILEGDBOUT_WRITE_OBJECT0
Beschreibung	Schreibt ein Objekt in die Datenbank. Das OUT-Objekt muss gemäss dem Objektmodell gefüllt sein.
Beispiel	<code>FILEGDBOUT_WRITE_OBJECT0</code>
Prozedur	FILEGDBOUT_WRITE_RECORD1 ! s recordname
Beschreibung	Schreibt ein Objekt definiert mit <recordname> in die Datenbank. <recordname> ist der Name einer Record-Definition (siehe weiter oben).
Beispiel	<code>... => FILEGDBOUT_WRITE_RECORD1,RECORD_1</code>
Prozedur	FILEGDBOUT_CLOSE [][[]]
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel

FILEGDBOUT_CLOSE

6.9. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,  
! erzeugt eine Tabelle für LFP's in der Datenbank  
! inklusive den Definitionen des Geodatabase Repository  
! und schreibt die LFP's in die Tabelle.
```

```
|LICENSE \license\iltoolspro.lic
```

```
MAP CONFIG_PARAM  
  GENERATE_MODEL => ON  
END_MAP
```

```
MAP USER_INPUT1  
  DIALOG      => FILE  
  MESSAGE     => 'Enter INTERLIS Input File'  
  FILE_FILTER => itf  
  FILE_EXISTS => TRUE  
  OPT        => input  
END_MAP
```

```
MAP USER_INPUT2  
  DIALOG      => DIRECTORY  
  MESSAGE     => 'Enter SDE or Database Output File'  
  OPT        => output  
END_MAP
```

```
MAP ILIN_PARAM  
  INTERLIS_DEF => '\models\DM01AVCH24LV95D.ili'  
  STATISTICS   => ON  
  CALC_SURFACE => ON  
  ENUM_TO_TEXT => ON  
  TRACE       => OFF  
END_MAP
```

```
MAP ILIN_TOPO  
  DEFAULT => OFF  
END_MAP
```

```
MAP FILEGDBOUT_PARAM  
  SRID          => 'CH1903+_LV95'  
  STATISTICS    => ON  
  DATASET       => ON  
END_MAP
```

```
MAP RECORD_1  
  TABLE => Fi_LFP1  
  GDB_DATASET => FixpunkteKategoriel  
  OBJID => CHAR(40),IN.OBJID  
  Entstehung => CHAR(40),IN.Entstehung.OBJID  
  NBIdent => CHAR(12),IN.NBIdent  
  Nummer => CHAR(12),IN.Nummer  
  SHAPE => GEOMETRY(point;2D;FALSE),IN.Geometrie  
  HoeheGeom => NUMBER(7,3),IN.HoeheGeom
```

```

LageGen => NUMBER(4,1),IN.LageGen
LageZuv => INTEGER,IN.LageZuv
LageZuv_TXT => CHAR(4),IN.LageZuv_TXT
HoeheGen => NUMBER(4,1),IN.HoeheGen
HoeheZuv => INTEGER,IN.HoeheZuv
HoeheZuv_TXT => CHAR(4),IN.HoeheZuv_TXT
Begehbarkeit => INTEGER,IN.Begehbarkeit
Begehbarkeit_TXT => CHAR(14),IN.Begehbarkeit_TXT
Punktzeichen => INTEGER,IN.Punktzeichen
Punktzeichen_TXT => CHAR(17),IN.Punktzeichen_TXT
END_MAP

MAP INPUT_SOURCES
  I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
  I1                                => IN.TOPIC,IN.TABLE
  I1,FixpunkteKategorie1,LFP1 => R_1,RECORD_1
  I1,*                              => OFF
END_MAP

|INCL \script\iltopo.mod
|INCL \script\filegdbout.mod
|INCL \script\il2filegdb\il2filegdb.out
|INCL \script\run1.prg

```

7. Modul GEOJSONOUT - GeoJSON schreiben

7.1. Allgemeines

Mit dem Modul GEOJSONOUT können Objekte in eine GeoJSON Datei geschrieben werden. Mit dem Modul können auch JSON Dateien geschrieben werden, die keine Erweiterungen für GeoJSON enthalten.

Der Modul GEOJSONOUT wird mit:

```
|INCL \script\geojsonout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

7.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

7.3. Parametermap GEOJSONOUT_PARAM

Folgende Parameter können in der Map SHPOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
CRS	o	EPSG:2056, Default: EPSG:2056 . Definiert das Koordinaten Referenz System..

STROKE_TOL	r	<real> . Stroke-Tolerance zur Auflösung von Kreisbögen in Liniensegmente. Als Stroke-Tolerance können dieselben Werte wie für die ICS-Methode <code>ICS.STROKE</code> verwendet werden (s.a. iG/Script Benutzer- und Referenzhandbuch). Mit <code>OFF</code> werden die Kreisbögen nicht aufgelöst.
STATISTICS	o	ON oder OFF, Default = OFF. Objektstatistik am Ender der .log Datei anzeigen.

7.4. Objektmodell

Allgemeine Komponenten für jedes OUT-Objekt

Komponente	req/opt	Beschreibung
OUT.GEOJSON_NAME(s)	o	Der Name des GeoJSON-Objektes.
OUT.GEOJSON_ID(s)	o	Die ID des GeoJSON-Objektes.
OUT.GEOJSON_GEOMETRY(g)	o	Die Geometrie des GeoJSON-Objektes.
OUT.<attribute>(*)	o	Weitere Objekte des JSON-Objektes.

7.5. Exportierte Prozeduren und Methoden

Prozedur `GEOJSONOUT_OPEN ! [s zipfile][[]]`

Beschreibung Öffnet den GEOJSONOUT Modul. Der Output wird in die ZIP-Datei geschrieben. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel `'c:\test.zip' GEOJSONOUT_OPEN`

Prozedur `GEOJSONOUT_WRITE_OBJECT0 ! [] []`

Beschreibung Schreibt ein Objekt. Für eine Erklärung der einzelnen Komponenten siehe Objektmodell.

Beispiel `GEOJSONOUT_WRITE_OBJECT0`

Prozedur `GEOJSONOUT_CLOSE ! [[]]`

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel `GEOJSONOUT_CLOSE`

7.6. Skriptbeispiel

```
! Diese ICS Konfiguration schreibt INTERLIS AV Daten in eine GeoJSON Datei
|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE
  MESSAGE => 'Enter .itf Input File'
```

```
FILE_FILTER => itf
FILE_EXISTS => TRUE
OPT => input
END_MAP

MAP USER_INPUT2
  DIALOG => FILE
  MESSAGE => 'Enter ZIP Output File'
  FILE_FILTER => zip
  FILE_EXISTS => FALSE
  OPT => output
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => ''
  LOG_TABLE => ON
  TRACE => OFF
  STATISTICS => ON
  CALC_SURFACE => ON
  ENUM_TO_TEXT => ON
  BUILD_DB => ON
END_MAP

MAP ILIN_TOPO
  Liegenschaften,Liegenschaft_Geometrie => AREA
  DEFAULT => OFF
END_MAP

MAP GEOJSONOUT_PARAM
  STATISTICS      => ON
  FENCE_MODE      => OFF
  STROKE_TOL      => 0.001
  CRS              => EPSG:2056
END_MAP

MAP INPUT_SOURCES
  I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Liegenschaften,Grundstueck => GEOJSONOUT_WRITE_OBJECT0
  I1,Liegenschaften,GrundstueckPos => GEOJSONOUT_WRITE_OBJECT0
  I1,Liegenschaften,Liegenschaft_Geometrie => GEOJSONOUT_WRITE_OBJECT0
  I1,Liegenschaften,Liegenschaft => GEOJSONOUT_WRITE_OBJECT0
  I1,Liegenschaften,Liegenschaft_Area => GEOJSONOUT_WRITE_OBJECT0
  I1,Liegenschaften,Liegenschaft_Boundary => GEOJSONOUT_WRITE_OBJECT0
  I1,* => OFF
END_MAP

PROCEDURE POST_READ_I1

  ! allgemeine Trigger-Procedure, die automatisch nach jedem Lesen der Input-Source I1 aufgerufen wird
  ! aus dem IN-Objekt wird das OUT-Objekt erstellt
  ! das OUT-Objekt wird dann geschrieben mit GEOJSONOUT_WRITE_OBJECT0

  &IN &OUT MAPCOPY
```

```
IF IN.TABLE '_Area' ENDS_WITH THEN
  &OUT 'Geometrie' MAPREM POP
END_IF

OUT.TOPIC . '_' . OUT.TABLE      => OUT.GEOJSON_NAME
OUT.GEOSHOP_ID                   => OUT.GEOJSON_ID

&OUT MAPRESET
WHILE &OUT MAPSCAN DO
  => LOCAL.NAME
  => LOCAL.VALUE

  IF LOCAL.VALUE IS_NULL THEN CONTINUE END_IF

  &LOCAL.VALUE GET_TYPE => LOCAL.TYPE

  IF LOCAL.TYPE = 'point'
    LOCAL.TYPE = 'line' OR
    LOCAL.TYPE = 'area' OR THEN

    &OUT 'GEOJSON_GEOMETRY' LOCAL.VALUE MAPINS
  END_IF

END_WHILE

END_PROCEDURE

MAP MACRO
END_MAP

| INCL \script\iltopodb.mod
| INCL \script\geojsonout.mod
| INCL \script\run1.prg
```

8. Modul GMMDBOUT - Intergraph GeoMedia ACCESS Datenbank schreiben

8.1. Allgemeines

Mit dem Skriptmodul können Objekte in eine ACCESS-Datenbank nach GeoMedia Spezifikationen geschrieben werden. Der Modul unterstützt speziell das GeoMedia Datenmodell und die GeoMedia Geometrien.

Der Modul beinhaltet folgende Besonderheiten:

- Schreibt Daten nach Access.
- Legt Tabellen für die Daten an.
- Schreibt die notwendigen Definitionen für GeoMedia.
- Schreibt die Geometrien für GeoMedia.
- Schreibt die GeoMedia Definitionen in das GeoMedia Repository.
- Die Datenbank kann nach dem Schreiben der Daten direkt mit GeoMedia verwendet werden.

Der Modul wird mit:

```
| INCL \script\gmmdbout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

8.2. Abhängigkeiten von anderen Modulen

Der Modul GMMDBOUT ist eine Erweiterung des Modules DBOUT. Alle in dem Modul DBOUT beschriebenen Anteile gelten daher auch für das Modul GMMDBOUT. Ziehen Sie deshalb die Dokumentation dieser Module bei.

8.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

8.4. Parametermap DBOUT_PARAM

Folgende Parameter können in der Map DBOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SQLTRACE	o	ON oder OFF, Default = OFF. Zeigt als sql-statements im Logfile an.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltet werden. Sie dazu mehr unter dem Kapitel Datasets. Der Wert ON hebt den Parameter DELETE_OLD auf.
SEED_DB	o	STRING. Definiert, eine Seed-Datenbank für die Output-Datenbank. Ist die Output-Datenbank nicht vorhanden, wird diese Seed-Datenbank in die Output-Datenbank vor dem Transfer kopiert. Die Seed-Datenbank beinhaltet bereits das GDB-Repository. Beispiel: c:\iltools\system\db\ESRI\GDB_seed_91.mdb. Siehe mehr dazu unter Modell Generierung.

8.5. Parametermap GMMDBOUT_PARAM

Folgende Parameter können in der Map GMMDBOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
CSGUID	r	STRING. Definiert das GUID des Koordinatensystems. Die definierte CSGUID muss in der Output-Datenbank in der GeoMedia-Tabelle GCoordSystem vorhanden sein. Beispiel: CSGUID => {ABD80B73-98D3-4537-8119-FBD238F2D703} für Schweizer Koordinatensystem.

Für die Anwendung von GeoMedia unter Access ist die entsprechende Dokumentation von INTERGRAPH zu beachten.

8.6. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tabellenname in welche das OUT-Objekt geschrieben werden soll.
OUT.<Attribut>(o)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Type des Wertes muss mit dem Type des Attributes in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden.

Dieses Objektmodell gilt für die Anwendung mit der Prozedur GMMDBOUT_WRITE_OBJECT0. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfiguration mit Prozeduren und Methoden vorgängig erzeugt werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, muss man das Kapitel mit den Record-Definitionen und die Prozedur GMMDBOUT_WRITE_RECORD1 beachten.

8.7. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank angelegt werden. Die Record Definitionen werden von der Prozedur GMMDBOUT_WRITE_RECORD1 verarbeitet. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<n>
  TABLE => <Table-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix RECORD_ beginnen und eine eindeutige Nummer <n> für die Record Definition beinhalten.

TABLE

Diese Komponente ist obligatorisch und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente TABLE die Datenbank-Tabelle.

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen innerhalb der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

CHAR(<length>)

Stringtypen sind immer als CHAR mit der Länge <length> zu definieren. Je nach Datenbank wird der Typ vom Modul in einen der Datenbank bekannten Type umgewandelt z.B. Oracle als VARCHAR2.

NUMBER

Number-Typen ohne Argumente werden vom Modul als NUMBER(38,5) interpretiert.

MEMO

Stringtyp für Texte > 255 Zeichen.

DATETIME

Datums/Zeit Typ. Der <Attribute-Value> muss der SQL-Spezifikation von MSACCESS entsprechen. Zum Beispiel für ein Datum <Attribute-Value> = '03.04.1993', für Datum/Zeit <Attribute-Value>='03.04.1993 17:34:00'.

GEONEDIA_GEOMETRY(<type>;<dimension>)

GeoMedia Geometrien müssen als Type GEONEDIA_GEOMETRY definiert werden. Dabei sind folgende weiteren Definitionen notwendig.

<type>

Type der Geometrie, einer der Werte: point|line|area|gmtext.

<dimension>

Dimension der Geometrie, einer der Werte 2D|3D. GeoMedia speichert Geometrien immer als 3D ab. Ist 2D definiert, so werden eventuelle Z-Koordinaten auf 0.0 gesetzt. Ist 3D definiert, so werden eventuell nicht vorhandene Z-Koordinaten auf 0.0 gesetzt.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können Konstanten oder ICS-Variablen, die einen Wert beinhalten - z.B. IN.OBJID - verwendet werden.

Beispiel einer Record Definition.

```
MAP RECORD_1
TABLE => Fixpunkte_LFP
OBJID => CHAR(10),IN.OBJID
ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
NUMMER => CHAR(12),IN.Nummer
GEOMETRIE => GEONEDIA_GEOMETRY(point;3D),IN.Geometrie
LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
BEGEHBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
SYMBOLORI => NUMBER,IN.SymbolOri
ART_TXT => CHAR(4),IN.Art_TXT
HERKUNFT => CHAR(30),IN.Herkunft
END_MAP
```

8.8. Datasets

Zur Verwaltung von Datasets in der Datenbank ist das analoge Kapitel des Modules DBOUT zu beachten.

8.9. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MODEL

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell erstellt werden soll, so ist der Parameter CONFIG_PARAM.GENERATE_MODEL auf ON zu setzen und das Script il2gmmdb.lib zu includen:

```
MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
:
| INCL \script\il2gdb\il2gmmdb.lib
```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell inklusive den Definitionen für das GeoMedia Repository generiert, falls es nicht schon generiert wurde.

In der Datenbank müssen die Tabellen des GeoMedia Repository bereits bestehen. Als leere GeoMedia Datenbank mit den Tabellen des GeoMedia Repository stehen folgende Datenbanken zur Verfügung:

```
ILTOOLS_DIR\system\db\GeoMedia\GeoMedia_seed.mdb
```

Kopieren Sie die gewünschte Datenbank für das Anlegen einer neuen GeoMedia-Datenbank oder erzeugen Sie mit GeoMedia eine neue leere GeoMedia-Datenbank.

Oder definieren Sie mit DBOUT_PARAM.SEED_DB eine Seed-Datenbank, die das GeoMedia Repository bereits beinhaltet.

Um das Datenmodell inklusive den Daten und den Definitionen im GeoMedia Repository aus einer GeoMedia Datenbank zu löschen, steht folgende Konfiguration zur Verfügung:

```
ILTOOLS_DIR\system\script\il2gmmdb\gmmdbdelete.cfg
```

Wird das Datenmodell im GeoMedia Repository durch die Schnittstelle generiert, ist folgendes vor Verwendung der Datenbank mit GeoMedia zu beachten.

8.10. Exportierte Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie der Modul DBOUT. Ziehen Sie deshalb die Dokumentation des Modules DBOUT bei.

Zusätzlich stellt der Modul GMMDBOUT folgende Prozeduren und Methoden zur Verfügung.

Prozedur	GMMDBOUT_OPEN [] []
Beschreibung	Öffnet eine Datenbank definiert mit DB_PARAM.SOURCE. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	GMMDBOUT_OPEN

Prozedur	GMMDBOUT_WRITE_OBJECT0
-----------------	------------------------

Beschreibung Schreibt einen Objekt in die Datenbank. Das OUT-Objekt muss gemäss dem Objektmodell vorbereitet sein.

Beispiel `GMMDBOUT_WRITE_OBJECT0`

Prozedur `GMMDBOUT_WRITE_RECORD1 ! s recordname`

Beschreibung Schreibt ein Objekt definiert in <recordname> in die Datenbank. <recordname> ist der Name einer Record-Definition (siehe weiter oben). Je nach den gesetzten Parametern führt die Prozedur beim ersten Aufruf für eine Record-Definition folgende Aktionen auf der Datenbank aus:

1. Die Tabelle oder deren Inhalt wird gelöscht, falls definiert durch Parameter.
2. Die Tabelle wird erzeugt, falls definiert durch Parameter.
3. Die GeoMedia Metadefinitionen werden generiert, falls definiert durch Parameter.
4. Das Objekt wird in die Datenbank geschrieben.

Beispiel `... => GMMDBOUT_WRITE_RECORD1,RECORD_1`

Prozedur `GMMDBOUT_CLOSE [][[]]`

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel `GMMDBOUT_CLOSE`

Methode `GEOMEDIA.GEOM_TO_BLOB [g geom][b blob]`

Beschreibung Übersetzt eine ICS-Geometrie point,line oder area in eine GeoMedia Geometrie als Blob. Falls die Geometrien nicht übersetzt werden können, wird auf dem Stack ein NULL-Blob zurückgegeben.

Beispiel `IN.GEOM GEOMEDIA.GEOM_TO_BLOB => VAR.GEOMEDIA_GEOM`

Folgende Konversionen werden durchgeführt:

point

to gdbPoint (10)

line

to gdbLinear (1)

area

to gdbAreal (2)

Für die Unterstützung des Produktes GeoMedia mit Access stehen folgende Prozeduren zur Verfügung.

Methode `GEOMEDIA.GEOM_TO_BLOB_POINT ! [p Pos, r Rot] [b blob]`

Beschreibung Erzeugt einen GeoMedia oriented Point.

Beispiel `IN.Geometrie IN.Ori GEOMEDIA.GEOM_TO_BLOB_POINT => VAR.GEOMEDIA_GEOM`

Methode `GEOMEDIA.GEOM_TO_BLOB_TEXT ! [s Text, p Pos, r Rot, i Hali, i Vali] [b blob]`

Beschreibung Erzeugt einen GeoMedia Text.

Beispiel `IN.Name IN.Geometrie IN.Ori IN.Hali IN.Vali GEOMEDIA.GEOM_TO_BLOB_TEXT => VAR.SDO_`

Prozedur `GMMDBOUT_POINT_CREATE2 ! p Pos, r Rot => IN.GMPoint`

Beschreibung Erzeugt einen GeoMedia oriented Point. Die GeoMedia Geometrie wird in IN.GMPoint abgelegt. IN.GMPoint kann in der Weiterverarbeitung genutzt werden, zum Beispiel in einer Record-Definition.

Beispiel `... => GMMDBOUT_POINT_CREATE3, IN.Geometrie, IN.Ori`

Prozedur `GMMDBOUT_TEXT_CREATE5 ! s Text, p Pos, r Rot, i Hali, i Vali => IN.GMText`

Beschreibung Erzeugt einen GeoMedia Text. Die GeoMedia Geometrie wird in IN.GMText abgelegt. IN.GMText kann in der Weiterverarbeitung genutzt werden, zum Beispiel in einer Record-Definition.

Beispiel `... => GMMDBOUT_TEXT_CREATE6, IN.Name, IN.Geometrie, IN.Ori, IN.HALI, IN.Vali`

Neben diesen Prozeduren des Moduls stehen auch die Prozeduren und Methoden des Moduls DBOUT zur Verfügung. Diese Prozeduren und Methoden sind im Modul DBOUT beschrieben.

8.11. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,
! erzeugt eine Tabelle für LFP's in der Datenbank
! inklusive den Definitionen des GEOMEDIA Repository
! und schreibt die LFP's in die Tabelle.
```

```
|LICENSE \license\iltoolspro.lic
```

```
MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
```

```
MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter INTERLIS Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT        => input
END_MAP
```

```
MAP USER_INPUT2
  DIALOG      => FILE
  MESSAGE     => 'Enter Access Output Database'
  FILE_FILTER => mdb
  FILE_EXISTS => FALSE
  OPT        => output
END_MAP
```

```
MAP ILIN_PARAM
  INTERLIS_DEF => \models\Grunddatensatz.ili
  STATISTICS  => ON
  CALC_SURFACE => ON
  ENUM_TO_TEXT => ON
```

```

TRACE          => OFF
END_MAP

MAP ILIN_TOPO
  DEFAULT => OFF
END_MAP

MAP DB_PARAM
  SOURCE       => '' ! ODBC-Source
  USER         => '' ! ODBC-User
  PASSWD       => '' ! ODBC-Password
  TRACE        => OFF
END_MAP

MAP DBOUT_PARAM
  STATISTICS   => ON
  DATASET      => ON
  SEED_DB      => \db\geomedia\geomedia_seed.mdb
END_MAP

MAP GMMDBOUT_PARAM
  CSGUID       => {ABD80B73-98D3-4537-8119-FBD238F2D703}
END_MAP

MAP RECORD_1
  TABLE => Fi_LFP
  OBJID  => CHAR(10),IN.OBJID
  ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
  NUMMER  => CHAR(12),IN.Nummer
  GEOMETRIE => GEOMEDIA_GEOMETRY(point;3D),IN.Geometrie
  GMTTEXT  => GEOMEDIA_GEOMETRY(gmtext;2D),IN.GMText
  LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
  HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
  BEGEBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
  SYMBOLORI => NUMBER,IN.SymbolOri
  ART_TXT  => CHAR(4),IN.Art_TXT
  HERKUNFT => CHAR(30),IN.Herkunft
END_MAP

MAP INPUT_SOURCES
  I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
  I1          => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => PR_1,GMMDBOUT_WRITE_RECORD1,RECORD_1
  I1,*       => OFF
END_MAP

MAP MACRO
  PR_1 => GMMDBOUT_TEXT_CREATE5,IN.Nummer,IN.NumPos,IN.NumOri,IN.NumHALi,IN.NumVALi
END_MAP

|INCL \script\iltopo.mod
|INCL \script\gmmdbout.mod
|INCL \script\il2gmddb\il2gmddb.lib
|INCL \script\run1.prg

```

9. Modul GMORAOUT - Intergraph GeoMedia Oracle Datenbank schreiben

9.1. Allgemeines

Mit dem Skriptmodul können Objekte in eine Oracle-Datenbank nach GeoMedia Spezifikationen via ODBC geschrieben werden. Das Modul unterstützt speziell die Oracle Spatial Option für räumliche Daten und die GeoMedia Metadaten in Oracle.

Der Modul beinhaltet folgende Besonderheiten:

- Schreibt Daten nach Oracle Spatial.
- Legt Tabellen für die Daten an.
- Schreibt die notwendigen Definitionen für Oracle Spatial.
- Schreibt die Geometrien für Oracle Spatial.
- Schreibt den Spatial Index für die Geometrien von Oracle Spatial.
- Schreibt die GEOMEDIA Definitionen in das GEOMEDIA Repository.
- Die Datenbank kann nach dem Schreiben der Daten direkt mit Oracle Spatial weiterbearbeitet werden.

Der Modul wird mit:

```
| INCL \script\gmoraout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

9.2. Abhängigkeiten von anderen Modulen

Der Modul GMORAOUT ist eine Erweiterung der Module DBOUT und ORAOUT . Alle in den Modulen DBOUT und ORAOUT beschriebenen Anteile gelten daher auch für das Modul GMORAOUT. Ziehen Sie deshalb die Dokumentation dieser Module bei.

9.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

9.4. Parametermap DBOUT_PARAM

Folgende Parameter können in der Map DBOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
CREATE_TABLE	o	ON, OFF, Default = OFF. Definiert ob die Tables definiert mit Record-Definitionen (siehe weiter unten) in der Datenbank erzeugt werden sollen. Mit OFF werden die Tabellen nicht erzeugt. Mit ON werden die Tabellen erzeugt und zwar zum Zeitpunkt, wenn das erste Objekt in die Tabelle geschrieben wird. Tables in die keine Objekte geschrieben werden, werden auch nicht erzeugt.
DELETE_OLD	o	ON, OFF oder DROP, Default = OFF. Löscht bestehende Daten in Tables definiert mit Record-Definitionen (siehe weiter unten). Mit OFF werden keine Daten gelöscht. Mit ON werden die Daten über ein delete-sql-statement gelöscht. Mit DROP wird die Table und damit die Daten gelöscht. Zusammen mit DROP und dem Parameter CREATE_TABLE => ON werden die Tables gelöscht und wieder erzeugt.
BATCH	o	ON oder OFF, Default = OFF. Mit OFF werden die sql-statements direkt auf der Datenbank ausgeführt. Mit OFF werden die sql-statements in ein Batchfile geschrieben. Mit dem Parameter BATCH_FILE wird das Batchfile definiert.
BATCH_OUTPUT_DIR	o	<directory>. Definiert ein Output-Directory für Batchfiles. Mit diesem Parameter kann das Output-Directory für Batchfiles definiert werden, falls der Parameter BATCH = ON definiert ist. Batchfiles können sein ein File mit SQL-Statementes oder in Kombination mit dem Oracle Output Modul die SQLLOADER-Bulkfiles. Ist dieser Parameter nicht gesetzt, so wird das Output-Directory aus einem eventuellen Input-File definiert in OPT.input bestimmt. Ist kein Input-File definiert, so ist das Output-Directory iltools\data\ics.sql.
BATCH_FILE	o	<file>. Definiert das Batchfile. Mit diesem Parameter kann das Batchfile definiert werden, falls der Parameter BATCH = ON definiert ist. Ist dieser Parameter nicht gesetzt, so wird das Batchfile aus einem eventuellen Input-File definiert in OPT.input mit der Endung .sql bestimmt. Ist kein Input-File definiert, so ist das Batchfile iltools\data\ics.sql als definiert. Das Batchfile beinhaltet SQL-Statements, um die transferierten Daten mittels SQL in eine Datenbank zu importieren.
SQLTRACE	o	ON oder OFF, Default = OFF. Zeigt als sql-statements im Logfile an.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltetet werden. Sie dazu mehr unter dem Kapitel Datasets. Der Wert ON hebt den Parameter DELETE_OLD auf.

9.5. Parametermap ORAOUT_PARAM

Folgende Parameter können in der Map ORAOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
-----------	---------	--------------

SQLLOADER_USE	o	ON oder OFF, Default = OFF. Definiert ob das Schreiben der Daten in die Datenbank über die Oracle-Utility SQLLoader erfolgen soll. SQLLoader ermöglicht das schnelle Laden grosser Datenmengen in Oracle Spatial. Ist dieser Parameter auf ON gesetzt, so werden sogenannte Bulk-Files für SQLLoader erstellt. Ist der Parameter DBOUT_PARAM.BATCH auf OFF gesetzt, so werden die Bulk-Files erstellt und mit SQLLoader während des Transfers in die Datenbank gelesen. Ist der Parameter DBOUT_PARAM.BATCH auf ON gesetzt, so werden lediglich die Bulk-Files erstellt. Diese Variante ermöglicht das Erstellen von Bulk-Files zur späteren Weiterverarbeitung oder zur Abgabe an Dritte.
SQLLOADER_CMD	o	<command>. Default undefiniert. Ist der Parameter ORAOUT_PARAM.SQLLOADER_USE auf ON gesetzt muss dieser Parameter mit dem Command für die Utility SQLLoader gesetzt sein. Das zu setzende Command entspricht dem Befehl, wie er auf einer Commandline für SQLLoader angewendet werden muss. Beispiel: sqlldr.exe userid=scott/tiger@ORACL. Anstatt des Commands kann im Parameter auch der Verweis auf eine Datei definiert werden, welche das Command beinhaltet.
SQLLOADER_DECIMAL-POINT	o	<char>. Default , . Definiert den Dezimalpunkt für reelle Zahlen in den SQLLoader-Bulk-Files.
SQLLOADER_CONTINUECODE	o	<string>. Default # . Definiert die Fortsetzungszeichen in den SQLLoader-Bulk-Files.
SQLLOADER_FIELDSEPARATOR	o	<string>. Default . Definiert die Spaltentrennzeichen in den SQLLoader-Bulk-Files.
SPATIAL_STROKE	o	<real> oder OFF, Default = OFF. Stroke-Tolerance zur Auflösung von Kreisbögen in Liniensegmente. Als Stroke-Tolerance können dieselben Werte wie für die ICS-Methode ICS.STROKE verwendet werden (s.a. iG/Script Benutzer- und Referenzhandbuch). Mit OFF werden die Kreisbögen nicht aufgelöst.
SPATIAL_METADATA	o	ON oder OFF, Default = ON. Definiert ob die Metadaten für Oracle Spatial in die Tabelle user_sdo_geom_metadata geschrieben werden sollen.
SPATIAL_INDEXCREATE	o	ON oder OFF, Default = ON. Definiert ob die Indexe für die Geometrie-Spalten für Oracle Spatial erzeugt werden sollen. Aufgrund der Tabelle und des Attributes wird automatisch ein Indexname erzeugt.
SPATIAL_INDEXDROP	o	ON oder OFF, Default = ON. Definiert ob die Indexe für die Geometrie-Spalten vor dem Schreiben der Daten gelöscht werden sollen. Dies ermöglicht das schnellere Schreiben der Daten. Mit ORAOUT_PARAM.SPATIAL_INDEXCREATE => ON werden die Indizes nach dem Schreiben wieder angelegt.
SPATIAL_INDEXTYPE	o	QTREE oder RTREE, Default = RTREE. Definiert den Indextyp für die Geometrie-Spalten von Oracle Spatial.
SPATIAL_INDEXTABLESPACE	o	<tablespace> oder OFF, Default = OFF. Definiert den Tablespace für die Indizes der Geometrie-Spalten von Oracle Spatial.
SPATIAL_VALIDATE	o	ON oder OFF, Default = OFF. Definiert ob Oracle Spatial SQL-Statements in das Logfile gesschrieben werden sollen. Mit diesen Statements können nachträglich unter Oracle Spatial die Geometrien validiert werden.

SPATIAL_SRID	o	<integer> oder OFF, Default = OFF. Definiert die ORacle-SRID-Identifikation für die Geometrien. Jede Geometrie wird mit dem definierten SRID nach Oracle geschrieben.
SPATIAL_META_X	o	<min,max,resolution>. Default 0,1000000,0.001. Definiert für die Metadaten von Oracle Spatial die Ausdehnung und Auflösung der X-Koordinaten.
SPATIAL_META_Y	o	<min,max,resolution>. Default 0,1000000,0.001. Definiert für die Metadaten von Oracle Spatial die Ausdehnung und Auflösung der Y-Koordinaten.
SPATIAL_META_Z	o	<min,max,resolution>. Default 0,1000000,0.001. Definiert für die Metadaten von Oracle Spatial die Ausdehnung und Auflösung der Z-Koordinaten.

Für die Anwendung von Oracle Spatial und der Oracle-Utility SQLLoader ist die entsprechende Dokumentation von Oracle zu beachten.

9.6. Parametermap GMORAOUT_PARAM

Folgende Parameter können in der Map GMORAOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
GDOSYS_PROCESS	o	ON oder OFF, Default = OFF. Definiert ob die GeoMedia Metadaten verarbeitet werden sollen. Mit OFF werden die GeoMedia Metadaten nicht verarbeitet. Mit ON werden die GeoMedia Metadaten verarbeitet. Der Oracle User GDOSYS und die erforderlichen Tabellen für die GeoMedia Metadaten müssen vorhanden sein.
GDOSYS_DELETE	o	ON oder OFF, Default = OFF. Definiert ob eventuell bestehende GeoMedia Metadaten zu den anzulegenden Objekten unter dem Oracle User GDOSYS vorgänig gelöscht werden sollen. Mit OFF werden die GeoMedia Metadaten nicht gelöscht. Mit ON werden die GeoMedia Metadaten gelöscht.
GDOSYS_CREATE	o	ON oder OFF, Default = OFF. Definiert ob die GeoMedia Metadaten angelegt werden sollen. Mit OFF werden die GeoMedia Metadaten nicht angelegt. Mit ON werden die GeoMedia Metadaten angelegt.
GDOSYS_SOURCE	o	<source>, <user>, <password>. Default ' '. Definiert die ODBC-Source mit User und Password für den User GDOSYS in der Oracle Datenbank.
GMADMIN_PROCESS	o	ON oder OFF, Default = OFF. Das Modul GMADMIN ist eine Erweiterung der INTERGRAPH Schweiz zur Verwaltung von Oracle-Users für den Zugriff auf die Oracle Daten unter GeoMedia. Das Modul bedingt zusätzliche Definitionen unter Oracle. Dieser Parameter definiert ob die GMADMIN Definitionen verarbeitet werden sollen. Mit OFF werden die GMADMIN Definitionen nicht verarbeitet. Mit ON werden die GMADMIN Definitionen verarbeitet. Der Oracle User GMADMIN muss vorhanden sein.
GMADMIN_DELETE	o	ON oder OFF, Default = OFF. Definiert ob eventuell bestehende GMADMIN Definitionen zu den anzulegenden Objekten vorgänig gelöscht werden sollen. Mit OFF werden die GMADMIN Definitionen nicht gelöscht. Mit ON werden die GMADMIN Definitionen gelöscht.
GMADMIN_CREATE	o	ON oder OFF, Default = OFF. Definiert ob die GMADMIN Definitionen angelegt werden sollen. Mit OFF werden die GMADMIN Definitionen nicht angelegt. Mit ON werden die GMADMIN Definitionen angelegt.

GMADMIN_SOURCE	o	<source>, <user>, <password>. Default ' '. Definiert die ODBC-Source mit User und Password für den User GMADMIN in der Oracle Datenbank.
PRIMARYKEY_PROCESS	o	ON oder OFF, Default = OFF. GeoMedia benötigt für jedes Objekt einen Primary Key. Dieser Parameter definiert, ob dieser Primary Key pro Tabelle angelegt und beim Schreiben der Daten unterhalten werden soll. Mit OFF wird der Primary Key nicht verarbeitet. Mit ON wird der Primary Key verarbeitet.
PRIMARYKEY_ATTRIBUTE	o	<Attribute-Name>, <Type>. Default = GM_ID, INTEGER. Definiert den Attributenamen und den Typ für den GeoMedia Primärschlüssel.
PRIMARYKEY_INDEXDROP	o	ON oder OFF, Default = OFF. Definiert, ob ein eventuell bestehender Index auf dem Primärschlüssel vor dem Schreiben der Daten gelöscht werden soll. Mit OFF wird der Index nicht gelöscht. Mit ON wird der Index gelöscht. Der Indexname wird vom Modul selber bestimmt.
PRIMARYKEY_INDEXCREATE	o	ON oder OFF, Default = OFF. Definiert, ob für den Primärschlüssel nach dem Schreiben der Daten ein Index angelegt werden soll. Mit OFF wird der Index nicht angelegt. Mit ON wird der Index angelegt. Der Indexname wird vom Modul selber bestimmt.
PRIMARYKEY_SEQUENCECREATE	o	ON oder OFF, Default = OFF. Definiert ob für den Primärschlüssel eine Sequence angelegt werden soll. Mit OFF wird die Sequence nicht angelegt. Mit ON wird die Sequence angelegt. Der Sequencename wird vom Modul selber bestimmt. Ist die Sequence bereits vorhanden, so wird sie mit den aktuellen Werten aktualisiert.

Für die Anwendung von GeoMedia unter Oracle ist die entsprechende Dokumentation von INTERGRAPH zu beachten.

9.7. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tabellenname in welche das OUT-Objekt geschrieben werden soll.
OUT.<Attribut>(o)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Type des Wertes muss mit dem Type des Attributes in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden.

Dieses Objektmodell gilt für die Anwendung mit der Prozedur GMORAOUT_WRITE_OBJECT0. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfiguration mit Prozeduren und Methoden vorgängig erzeugt werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, muss man das Kapitel mit den Record-Definitionen und die Prozedur GMORAOUT_WRITE_RECORD1 beachten.

9.8. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank angelegt werden. Die Record Definitionen werden von der Prozedur GMORAOUT_WRITE_RECORD1 verarbeitet. Eine Record Definition sieht wie folgt aus.

```

MAP RECORD_<n>
  TABLE => <Table-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
  :
END_MAP

```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix RECORD_ beginnen und eine eindeutige Nummer <n> für die Record Definition beinhalten.

TABLE

Diese Komponente ist obligatorisch und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente TABLE die Datenbank-Tabelle.

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen innerhalb der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

CHAR(<length>)

Stringtypen sind immer als CHAR mit der Länge <length> zu definieren. Je nach Datenbank wird der Typ vom Modul in einen der Datenbank bekannten Type umgewandelt z.B. Oracle als VARCHAR2.

NUMBER

Number-Typen ohne Argumente werden vom Modul als NUMBER(38,5) interpretiert.

MDSYS.SDO_GEOMETRY(<type>;<dimension>;<resolution>)

Oracle Spatial Geometrien müssen als Type MDSYS.SDO_GEOMETRY definiert werden. Dabei sind folgende weiteren Definitionen notwendig.

<type>

Type der Geometrie, einer der Werte: point|line|area|gmtext.

<dimension>

Dimension der Geometrie, einer der Werte: 2D|3D.

<resolution>

Real-Wert der Auflösung.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können Konstanten oder ICS-Variablen, die einen Wert beinhalten - z.B. IN.OBJID - verwendet werden.

Beispiel einer Record Definition.

```

MAP RECORD_1
  TABLE => Fixpunkte_LFP
  OBJID => CHAR(10),IN.OBJID
  ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
  NUMMER => CHAR(12),IN.Nummer

```

```

GEOMETRIE => MDSYS.SDO_GEOMETRY(point;3D;0.001),IN.Geometrie
LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
BEGEHBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
SYMBOLORI => NUMBER,IN.SymbolOri
ART_TXT => CHAR(4),IN.Art_TXT
HERKUNFT => CHAR(30),IN.Herkunft
END_MAP

```

9.9. Datasets

Zur Verwaltung von Datasets in der Datenbank ist das analoge Kapitel des Modules DBOUT zu beachten.

9.10. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MODEL

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell inklusive den notwendigen GEOMEDIA Definitionen erstellt werden soll, so ist der Parameter CONFIG_PARAM.GENERATE_MODEL auf ON zu setzen und das Script `il2gmora.lib` zu includen:

```

MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
:
|INCL \script\il2gmora\il2gmora.lib

```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell angelegt, falls es nicht schon angelegt wurde.

In der Datenbank müssen die Tabellen des GEOMEDIA Repository bereits bestehen.

Um das Datenmodell inklusive den Daten und den Definitionen im GEOMEDIA Repository aus einer Datenbank zu löschen, steht folgende Konfiguration zur Verfügung:

```

ILTOOLS_DIR\system\script\il2gmora\gmoradelete.cfg

```

9.11. Exportierte Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie der Modul DBOUT. Ziehen Sie deshalb die Dokumentation des Modules DBOUT bei.

Zusätzlich stellt der Modul GMORAOUT folgende Prozeduren und Methoden zur Verfügung.

Prozedur	<code>GMORAOUT_OPEN [][]</code>
Beschreibung	Öffnet eine Datenbank definiert mit <code>DB_PARAM.SOURCE</code> . Die Prozedur wird von <code>RUN1</code> automatisch aufgerufen.
Beispiel	<code>GMORAOUT_OPEN</code>
Prozedur	<code>GMORAOUT_WRITE_OBJECT0</code>
Beschreibung	Schreibt einen Objekt in die Datenbank. Das OUT-Objekt muss gemäss dem Objektmodell vorbereitet sein.
Beispiel	<code>GMORAOUT_WRITE_OBJECT0</code>

Prozedur	<code>GMORAOUT_WRITE_RECORD1 ! s recordname</code>
Beschreibung	<p>Schreibt ein Objekt definiert in <recordname> in die Datenbank. <recordname> ist der Name einer Record-Definition (siehe weiter oben). Je nach den gesetzten Parametern führt die Prozedur beim ersten Aufruf für eine Record-Definition folgende Aktionen auf der Datenbank aus:</p> <ol style="list-style-type: none"> 1. Die Tabelle oder deren Inhalt wird gelöscht, falls definiert durch Parameter. 2. Die Tabelle wird erzeugt, falls definiert durch Parameter. 3. Die Oracle Spatial Metadefinitionen werden generiert, falls definiert durch Parameter. 4. Die GEOMEDIA Metadefinitionen werden generiert, falls definiert durch Parameter. 5. Das Objekt wird in die Datenbank geschrieben.
Beispiel	<code>... => GMORAOUT_WRITE_RECORD1 ,RECORD_1</code>
Prozedur	<code>GMORAOUT_CLOSE [] []</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>GMORAOUT_CLOSE</code>
Methode	<code>ORACLE.GEOM_TO_SDO_GEOMETRY [g li geom, i dimension][s sdo-geometry]</code>
Beschreibung	Übersetzt eine ICS-Geometrie point,line oder area in eine Oracle-Spatial Geometrie als String. Als Input können auch Listen von Geometrien übergeben werden. Falls die Geometrien nicht übersetzt werden können, wird auf dem Stack ein NULL-String zurückgegeben.
Beispiel	<code>IN.GEOM 2 ORACLE.GEOM_TO_SDO_GEOMETRY => VAR.SDO_GEOM</code>
	Folgende Konversionen werden durchgeführt:
	point to SDO-point
	line to SDO-line
	area to SDO-polygon
	list of points to SDO-multipoints
	list of lines to SDO-multilines
	list of areas to SDO-multipolygon
	list of points and/or lines and/or areas to SDO-collection

Für die Unterstützung des Produktes GeoMedia mit Oracle Spatial stehen folgende Prozeduren zur Verfügung.

Methode	<code>ORACLE.GEOM_TO_SDO_GEOMETRY_ORIENTEDPOINT ! [p Pos, r Rot, i Dimension] [s sdo_geometry]</code>
Beschreibung	Erzeugt eine Oracle Spatial Geometry als oriented Point.
Beispiel	<pre>IN.Geometrie IN.Ori 2 ORACLE.GEOM_TO_SDO_GEOMETRY_ORIENTEDPOINT => VAR.SDO_GEOM</pre>
Methode	<code>ORACLE.GEOM_TO_SDO_GEOMETRY_GEOMEDIA_POINT ! [p Pos, r Rot, i Dimension] [s sdo_geometry]</code>
Beschreibung	Erzeugt eine Oracle Spatial Geometry als GeoMedia Point.
Beispiel	<pre>IN.Geometrie IN.Ori 2 ORACLE.GEOM_TO_SDO_GEOMETRY_GEOMEDIA_POINT => VAR.SDO_GEOM</pre>
Methode	<code>ORACLE.GEOM_TO_SDO_GEOMETRY_GEOMEDIA_TEXT ! [s Text, p Pos, r Rot, i Hali, i Vali , i Dimension] [s sdo_geometry]</code>
Beschreibung	Erzeugt eine Oracle Spatial Geometry als GeoMedia Text.
Beispiel	<pre>IN.Name IN.Geometrie IN.Ori IN.Hali IN.Vali 2 ORACLE.GEOM_TO_SDO_GEOMETRY_GEOMEDIA_TEXT => VAR.SDO_GEOM</pre>
Prozedur	<code>GMORAOUT_POINT_CREATE3 ! p Pos, r Rot, i Dimension => IN.GMPoint</code>
Beschreibung	Erzeugt eine Oracle Spatial Geometry als GeoMedia Point. Die Oracle Spatial Geometry wird in IN.GMPoint abgelegt. IN.GMPoint kann in der Weiterverarbeitung genutzt werden, zum Beispiel in einer Record-Definition.
Beispiel	<pre>... => GMORAOUT_POINT_CREATE3, IN.Geometrie, IN.Ori, 2</pre>
Prozedur	<code>GMORAOUT_TEXT_CREATE6 ! s Text, p Pos, r Rot, i Hali, i Vali , i Dimension => IN.GMText</code>
Beschreibung	Erzeugt eine Oracle Spatial Geometry als GeoMedia Text. Die Oracle Spatial Geometry wird in IN.GMText abgelegt. IN.GMText kann in der Weiterverarbeitung genutzt werden, zum Beispiel in einer Record-Definition.
Beispiel	<pre>... => GMORAOUT_TEXT_CREATE6, IN.Name, IN.Geometrie, IN.Ori, IN.Hali, IN.Vali, 2</pre>

Neben diesen Prozeduren des Moduls stehen auch die Prozeduren und Methoden des Moduls DBOUT zur Verfügung. Diese Prozeduren und Methoden sind im Modul DBOUT beschrieben.

9.12. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,
! erzeugt eine Tabelle für LFP's in der Datenbank
! inklusive den Definitionen des GEOMEDIA Repository
! und schreibt die LFP's in die Tabelle.
```

```
|LICENSE \license\iltoolspro.lic
```

```
MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
```

```

MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter INTERLIS Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT         => input
END_MAP

MAP USER_INPUT2
  DIALOG      => ODBC
  OPT         => output
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\Grunddatensatz.ili
  STATISTICS   => ON
  CALC_SURFACE => ON
  ENUM_TO_TEXT => ON
  TRACE        => OFF
END_MAP

MAP ILIN_TOPO
  DEFAULT     => OFF
END_MAP

MAP DB_PARAM
  SOURCE      => '' ! ODBC-Source
  USER        => '' ! ODBC-User
  PASSWD      => '' ! ODBC-Password
  TRACE       => OFF
END_MAP

MAP DBOUT_PARAM
  STATISTICS   => ON
  CREATE_TABLE => ON
  DELETE_OLD   => DROP
  BATCH        => OFF
END_MAP

MAP ORAOUT_PARAM
  SPATIAL_STROKE      => OFF ! OFF or a real Stroke-Tolerance
  SPATIAL_METAINSERT  => ON  ! ON|OFF Spatial Meta Insert
  SPATIAL_INDEXDROP   => ON  ! ON|OFF Spatial Index Drop
                        ! before Insert
  SPATIAL_INDEXCREATE => ON  ! ON|OFF Spatial Index Create
                        ! after Insert
  SPATIAL_VALIDATE    => OFF ! ON|OFF Spatial Validate after Insert
  SPATIAL_META_X      => 0,1000000,0.001 ! x-min,x-max,x-tolerance
  SPATIAL_META_Y      => 0,1000000,0.001 ! y-min,y-max,y-tolerance
  SPATIAL_META_Z      => 0,1000000,0.001 ! z-min,z-max,z-tolerance
  SPATIAL_INDEXTABLESPACE => OFF
  SPATIAL_SRID        => OFF
END_MAP

MAP GMORAOUT_PARAM
  GDOSYS_PROCESS      => ON
  GDOSYS_DELETE       => ON

```

```
GDOSYS_CREATE      => ON
GDOSYS_SOURCE      => GDOSYS ,GDOSYS ,GDOSYS

GMADMIN_PROCESS    => ON
GMADMIN_DELETE     => ON
GMADMIN_CREATE     => ON
GMADMIN_SOURCE     => GMADMIN ,GMADMIN ,GMADMIN

PRIMARYKEY_PROCESS => ON
PRIMARYKEY_ATTRIBUTE => GM_ID ,INTEGER
PRIMARYKEY_INDEXDROP => OFF
PRIMARYKEY_INDEXCREATE => OFF
PRIMARYKEY_SEQCREATE => ON
END_MAP

MAP RECORD_1
TABLE => Fi_LFP
OBJID => CHAR(10) ,IN.OBJID
ENTSTEHUNG => CHAR(10) ,IN.Entstehung.OBJID
NUMMER => CHAR(12) ,IN.Nummer
GEOMETRIE => MDSYS.SDO_GEOMETRY(point;3D;0.001) ,IN.Geometrie
GMTTEXT => MDSYS.SDO_GEOMETRY(gmtext;2D;0.001) ,IN.GMText
LAGEZUV_TXT => CHAR(4) ,IN.LageZuv_TXT
HOEHEZUV_TXT => CHAR(4) ,IN.HoeheZuv_TXT
BEGEHBARKEIT_TXT => CHAR(14) ,IN.Begehbarkeit_TXT
SYMBOLORI => NUMBER ,IN.SymbolOri
ART_TXT => CHAR(4) ,IN.Art_TXT
HERKUNFT => CHAR(30) ,IN.Herkunft
END_MAP

MAP INPUT_SOURCES
I1 => ILTOPO ,OPT.input
END_MAP

MAP INOUT
I1          => IN.TOPIC ,IN.TABLE
I1,Fixpunkte,LFP => PR_1 ,GMORAOUT_WRITE_RECORD1 ,RECORD_1
I1,*       => OFF
END_MAP

MAP MACRO
PR_1 => GMORAOUT_TEXT_CREATE6 ,IN.Nummer ,IN.NumPos ,IN.NumOri ,IN.NumHali ,IN.NumVali ,2
END_MAP

| INCL \script\iltopo.mod
| INCL \script\gmoraout.mod
| INCL \script\il2gmora\il2gmora.lib
| INCL \script\run1.prg
```

10. Modul GMSQLOUT - Intergraph GeoMedia SQL Server Datenbank schreiben

10.1. Allgemeines

Mit dem Skriptmodul können Objekte in eine SQL Server-Datenbank nach GeoMedia Spezifikationen geschrieben werden. Der Modul unterstützt speziell das GeoMedia Datenmodell, die GeoMedia Geometrien, das SQL Server Datenmodell und die SQL Server Geometrien.

Der Modul beinhaltet folgende Besonderheiten:

- Schreibt Daten nach SQL Server.
- Legt Tabellen für die Daten an.
- Schreibt die GeoMedia Definitionen in das GeoMedia Repository.
- Schreibt die Geometrien für GeoMedia.
- Schreibt die SQL Server Definitionen (optional).
- Schreibt die Geometrien für SQL Server.
- Die Datenbank kann nach dem Schreiben der Daten direkt mit GeoMedia verwendet werden.

Der Modul wird mit:

```
|INCL \script\gmsqlout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

10.2. Abhängigkeiten von anderen Modulen

Der Modul GMSQLOUT ist eine Erweiterung des Modules DBOUT. Alle in dem Modul DBOUT beschriebenen Anteile gelten daher auch für das Modul GMSQLOUT. Ziehen Sie deshalb die Dokumentation dieser Module bei.

10.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

10.4. Parametermap DBOUT_PARAM

Folgende Parameter können in der Map DBOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltet werden. Sie dazu mehr unter dem Kapitel Datasets. Der Wert ON hebt den Parameter DELETE_OLD auf.

10.5. Parametermap GMSQLOUT_PARAM

Folgende Parameter können in der Map GMSQLOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SRID	o	STRING. Definiert den Namen des Koordinatensystems. Das Koordinatensystems muss in der Output-Datenbank in der GeoMedia-Tabelle GCoordSystem vorhanden sein. Beispiel: SRID => 'LV03 - CH1903' für Schweizer Koordinatensystem. Falls der Parameter nicht definiert ist, wird mit GParameters.GPARAMETER='DefaultCoordinateSystem' das Koordinatensystem bestimmt. Die Schweizer Koordinatensystems können Sie mit den GeoMedia Database Utilities aus folgende Files in die Datenbank einlesen: ILTOOLS_DIR\system\db\GeoMedia*.cfs
PRIMARYKEY_ATTRIBUTE	o	STRING. Definiert den Attributnamen mit dem eindeutigen Schlüssel für die GeoMedia Objekte.
SPATIAL_GEOM_NATIVE	o	ON oder OFF, Default = ON. Definiert, ob die Geometrien neben den GeoMedia Geometrien auch als SQL Server Geometrien geschrieben werden. Wenn ON entsteht zu einem GeoMedia Geomtrie Attribut <geometry> ein weiteres SQL Server Geometrie Attribut <geometry>_SPA.
SPATIAL_GEOM_CLEAN	o	ON oder OFF, Default =OFF. Definiert, ob die Geometrien gecleant werden.

Für die Anwendung von GeoMedia unter SQL Server ist die entsprechende Dokumentation von INTERGRAPH zu beachten.

10.6. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tabellenname in welche das OUT-Objekt geschrieben werden soll.
OUT.<Attribut>(o)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Type des Wertes muss mit dem Type des Attributes in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden.

Dieses Objektmodell gilt für die Anwendung mit der Prozedur GMSQLOUT_WRITE_OBJECT0. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfigu-

ration mit Prozeduren und Methoden vorgängig erzeugt werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, muss man das Kapitel mit den Record-Definitionen und die Prozedur `GMSQLOUT_WRITE_RECORD1` beachten.

10.7. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank angelegt werden. Die Record Definitionen werden von der Prozedur `GMSQLOUT_WRITE_RECORD1` verarbeitet. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<n>
  TABLE => <Table-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix `RECORD_` beginnen und eine eindeutige Nummer `<n>` für die Record Definition beinhalten.

TABLE

Diese Komponente ist obligatorisch und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente `TABLE` die Datenbank-Tabelle.

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen innerhalb der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

CHAR(<length>)

Stringtypen sind immer als `CHAR` mit der Länge `<length>` zu definieren. Je nach Datenbank wird der Typ vom Modul in einen der Datenbank bekannten Type umgewandelt z.B. Oracle als `VARCHAR2`.

NUMBER

Number-Typen ohne Argumente werden vom Modul als `NUMBER(38,5)` interpretiert.

GEONEDIA_GEOMETRY(<type>;<dimension>)

GeoMedia Geometrien müssen als Type `GEONEDIA_GEOMETRY` definiert werden. Dabei sind folgende weiteren Definitionen notwendig.

<type>

Type der Geometrie, einer der Werte: `point|line|area|gmtext`.

<dimension>

Dimension der Geometrie, einer der Werte `2D|3D`. GeoMedia speichert Geometrien immer als `3D` ab. Ist `2D` definiert, so werden eventuelle Z-

Kordinaten auf 0.0 gesetzt. Ist 3D definiert, so werden eventuell nicht vorhandene Z-Koordinaten auf 0.0 gesetzt.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können Konstanten oder ICS-Variablen, die einen Wert beinhalten - z.B. IN.OBJID - verwendet werden.

Beispiel einer Record Definition.

```
MAP RECORD_1
  TABLE => Fixpunkte_LFP
  OBJID => CHAR(10),IN.OBJID
  ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
  NUMMER => CHAR(12),IN.Nummer
  GEOMETRIE => GEOMEDIA_GEOMETRY(point;3D),IN.Geometrie
  LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
  HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
  BEGEHBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
  SYMBOLORI => NUMBER,IN.SymbolOri
  ART_TXT => CHAR(4),IN.Art_TXT
  HERKUNFT => CHAR(30),IN.Herkunft
END_MAP
```

10.8. Datasets

Zur Verwaltung von Datasets in der Datenbank ist das analoge Kapitel des Modules DBOUT zu beachten.

10.9. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MODEL

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell erstellt werden soll, so ist der Parameter CONFIG_PARAM.GENERATE_MODEL auf ON zu setzen und das Script il2gmsql.lib zu includen:

```
MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
:
|INCL \script\il2gdb\il2gmsql.lib
```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell inklusive den Definitionen für das GeoMedia Repository generiert, falls es nicht schon generiert wurde.

In der Datenbank müssen die Tabellen des GeoMedia Repository bereits bestehen. Erstellen Sie das GeoMedia Repository mit den GeoMedia Database Utilities.

Um das Datenmodell inklusive den Daten und den Definitionen im GeoMedia Repository aus einer GeoMedia Datenbank zu löschen, steht folgende Konfiguration zur Verfügung:

```
ILTOOLS_DIR\system\script\il2gmsql\gmsqldelete.cfg
```

10.10. Exportierte Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie der Modul DBOUT. Ziehen Sie deshalb die Dokumentation des Modules DBOUT bei.

Zusätzlich stellt der Modul GMSQLOUT folgende Prozeduren und Methoden zur Verfügung.

Prozedur `GMSQLOUT_OPEN [] []`
Beschreibung Öffnet eine Datenbank definiert mit `DB_PARAM.SOURCE`. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel `GMSQLOUT_OPEN`

Prozedur `GMSQLOUT_WRITE_OBJECT0`
Beschreibung Schreibt einen Objekt in die Datenbank. Das OUT-Objekt muss gemäss dem Objektmodell vorbereitet sein.

Beispiel `GMSQLOUT_WRITE_OBJECT0`

Prozedur `GMSQLOUT_WRITE_RECORD1 ! s recordname`
Beschreibung Schreibt ein Objekt definiert in `<recordname>` in die Datenbank. `<recordname>` ist der Name einer Record-Definition (siehe weiter oben). Je nach den gesetzten Parametern führt die Prozedur beim ersten Aufruf für eine Record-Definition folgende Aktionen auf der Datenbank aus:

1. Die Tabelle oder deren Inhalt wird gelöscht, falls definiert durch Parameter.
2. Die Tabelle wird erzeugt, falls definiert durch Parameter.
3. Die GeoMedia Metadefinitionen werden generiert, falls definiert durch Parameter.
4. Das Objekt wird in die Datenbank geschrieben.

Beispiel `... => GMSQLOUT_WRITE_RECORD1,RECORD_1`

Prozedur `GMSQLOUT_CLOSE [] []`
Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel `GMSQLOUT_CLOSE`

Methode `GEOMEDIA.GEOM_TO_BLOB [g geom][b blob]`
Beschreibung Übersetzt eine ICS-Geometrie point,line oder area in eine GeoMedia Geometrie als Blob. Falls die Geometrien nicht übersetzt werden können, wird auf dem Stack ein NULL-Blob zurückgegeben.

Beispiel `IN.GEOM GEOMEDIA.GEOM_TO_BLOB => VAR.GEOMEDIA_GEOM`

Folgende Konversionen werden durchgeführt:

point
to gdbPoint (10)

line
to gdbLinear (1)

area
to gdbAreal (2)

Für die Unterstützung des Produktes GeoMedia mit SQL Server stehen folgende Prozeduren zur Verfügung.

Methode	<code>GEOMEDIA.GEOM_TO_BLOB_POINT ! [p Pos, r Rot] [b blob]</code>
Beschreibung	Erzeugt einen GeoMedia oriented Point.
Beispiel	<code>IN.Geometrie IN.Ori GEOMEDIA.GEOM_TO_BLOB_POINT => VAR.GEOMEDIA_GEOM</code>
Methode	<code>GEOMEDIA.GEOM_TO_BLOB_TEXT ! [s Text, p Pos, r Rot, i Hali, i Vali] [b blob]</code>
Beschreibung	Erzeugt einen GeoMedia Text.
Beispiel	<code>IN.Name IN.Geometrie IN.Ori IN.Hali IN.Vali GEOMEDIA.GEOM_TO_BLOB_TEXT => VAR.SDO_</code>
Prozedur	<code>GMSQLOUT_POINT_CREATE2 ! p Pos, r Rot => IN.GMPoint</code>
Beschreibung	Erzeugt einen GeoMedia oriented Point. Die GeoMedia Geometrie wird in IN.GMPoint abgelegt. IN.GMPoint kann in der Weiterverarbeitung genutzt werden, zum Beispiel in einer Record-Definition.
Beispiel	<code>... => GMSQLOUT_POINT_CREATE3, IN.Geometrie, IN.Ori</code>
Prozedur	<code>GMSQLOUT_TEXT_CREATE5 ! s Text, p Pos, r Rot, i Hali, i Vali => IN.GMText</code>
Beschreibung	Erzeugt einen GeoMedia Text. Die GeoMedia Geometrie wird in IN.GMText abgelegt. IN.GMText kann in der Weiterverarbeitung genutzt werden, zum Beispiel in einer Record-Definition.
Beispiel	<code>... => GMSQLOUT_TEXT_CREATE6, IN.Name, IN.Geometrie, IN.Ori, IN.HALI, IN.Vali</code>

Neben diesen Prozeduren des Moduls stehen auch die Prozeduren und Methoden des Moduls DBOUT zur Verfügung. Diese Prozeduren und Methoden sind im Modul DBOUT beschrieben.

10.11. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,
! erzeugt eine Tabelle für LFP's in der Datenbank
! inklusive den Definitionen des GEOMEDIA Repository
! und schreibt die LFP's in die Tabelle.
```

```
|LICENSE \license\iltoolspro.lic
```

```
MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
```

```
MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter INTERLIS Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT         => input
END_MAP
```

```
MAP USER_INPUT2
  DIALOG      => ODBC_FILE
```

```
MESSAGE      => 'Enter ODBC or Database Output File'
OPT          => output
END_MAP

MAP ILIN_PARAM
INTERLIS_DEF => \models\Grunddatensatz.ili
STATISTICS  => ON
CALC_SURFACE => ON
ENUM_TO_TEXT => ON
TRACE       => OFF
END_MAP

MAP ILIN_TOPO
DEFAULT     => OFF
END_MAP

MAP DB_PARAM
SOURCE      => '' ! ODBC-Source
USER        => '' ! ODBC-User
PASSWD     => '' ! ODBC-Password
TRACE       => OFF
END_MAP

MAP DBOUT_PARAM
STATISTICS  => ON
DATASET     => ON
SEED_DB     => \db\geomedia\geomedia_seed.mdb
END_MAP

MAP GMSQLOUT_PARAM
SRID        => 'LV03 - CH1903'
PRIMARYKEY_ATTRIBUTE => GM_ID
SPATIAL_GEOM_NATIVE  => ON
SPATIAL_GEOM_CLEAN   => ON
END_MAP

MAP RECORD_1
TABLE      => Fi_LFP
OBJID     => CHAR(10),IN.OBJID
ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
NUMMER    => CHAR(12),IN.Nummer
GEOMETRIE => GEOMEDIA_GEOMETRY(point;3D),IN.Geometrie
GMTEXT    => GEOMEDIA_GEOMETRY(gmtext;2D),IN.GMText
LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
BEGEHBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
SYMBOLORI => NUMBER,IN.SymbolOri
ART_TXT   => CHAR(4),IN.Art_TXT
HERKUNFT  => CHAR(30),IN.Herkunft
END_MAP

MAP INPUT_SOURCES
I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
I1          => IN.TOPIC,IN.TABLE
I1,Fixpunkte,LFP => PR_1,GMSQLOUT_WRITE_RECORD1,RECORD_1
```

```

I1,*          => OFF
END_MAP

MAP MACRO
  PR_1 => GMSQLOUT_TEXT_CREATE5,IN.Nummer,IN.NumPos,IN.NumOri,IN.NumHali,IN.NumVali
END_MAP

|INCL \script\iltopo.mod
|INCL \script\gmsqlout.mod
|INCL \script\il2sqlmdb\il2gmsql.lib
|INCL \script\run1.prg

```

11. Modul IFCOUT - Industry Foundation Classes IFC schreiben

11.1. Allgemeines

Mit dem Modul können ICS Objekte nach IFC-Files - STEP Physical File (SPF) *.ifc - geschrieben werden.

Das Modul basiert auf dem IFC SDK der Open Design Alliance (ODA).

Der Modul wird mit:

```
|INCL \script\ODAIFCOUT.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

11.2. Parametermap IFCOUT_PARAM

Folgende Parameter können in der Map IFCOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
VERSION	o	IFC2x3 IFC4. IFC Version des Outputfiles. Default ist IFC4 .
GEOM_ORIGIN	o	<x>, <y>. Geometrie-Origin zu dem die Geometrien der Objekte relativ mit einem Offset geschrieben werden.
STROKE	o	<real>. Strokewert mit dem Kreisbögen in Liniensegmente aufgelöst werden.
STATISTICS	o	ON oder OFF. Default OFF. Statistik anzeigen ein oder aus.

11.3. Objektmodell

Für das Schreiben mit dem Modul können im OUT-Objekt folgende Komponenten gesetzt werden:

Allgemeine Komponenten für jedes OUT-Objekt

Komponente	req/opt	Beschreibung
OUT.IFC_CLASS(s)	o	IFC Class des Objektes. Ist die Klasse nicht definiert, wird eine Klasse IFCBUILDINGELEMENTPROXY geschrieben.
OUT.IFC_NAME(s)	o	Name des Objektes.

OUT.IFC_TAG(s)	o	Tag des Objektes.
OUT.IFC_GEOM(g,li,m)	o	Geometrie des Objektes. (siehe unten)
OUT.IFC_PROFI- LE(s)	o	CIRCLE ELLIPSE RECTANGLE. Profilname für die Geometrie des Objektes.
OUT.IFC_GEOM_LENGTH(r)	o	Länge des Objektes. Mit Profil, die Länge des Profiles.
OUT.IFC_GEOM_WIDTH(r)	o	Breite des Objektes. Mit Profil, die Breite des Profiles.
OUT.IFC_GEOM_HEIGHT(r)	o	Höhe des Objektes. Mit Profil, die Höhe des Profiles.
OUT.IFC_GEOM_XSCALE(r)	o	Die Skalierung des Objektes in die x-Richtung.
OUT.IFC_GEOM_YSCALE(r)	o	Die Skalierung des Objektes in die y-Richtung.
OUT.IFC_GEOM_ZSCALE(r)	o	Die Skalierung des Objektes in die z-Richtung.
OUT.IFC_GEOM_XOFFSET(r)	o	Ein Offset des Objektes in die x-Richtung.
OUT.IFC_GEOM_YOFFSET(r)	o	Ein Offset des Objektes in die y-Richtung.
OUT.IFC_GEOM_ZOFFSET(r)	o	Ein Offset des Objektes in die z-Richtung.
OUT.IFC_GEOM_ROTATION(r)	o	Ein Drehwinkel des Objektes.
OUT.IFC_COLOR(s)	o	R/G/B . Farbe des Objektes. z.B. 125/233/52
OUT.IFC_BUILDINGELEMENTPROXYTYPE(s)	o	File mit absolem oder relativen Pfad mit einem Objekttyp zum inkludieren im Output-File.
OUT.IFC_PROPERTYSETS(li)	o	Liste von Propertysets des Objektes. Jedes Propertyset ist eine Map.
OUT.*(*)	o	Weitere Attribute zur IFC-Klasse, wie diese in der Beschreibung der IFC Klassen aufgeführt sind. Siehe zum Beispiel https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/

Inhalt für OUT.IFC_GEOM

OUT.IFC_GEOM kann folgendes beinhalten:

Inhalt	Beschreibung
OUT.IFC_GEOM(g)	Eine einzelne Geometrie
OUT.IFC_GEOM(li of g)	Eine Liste von Geometrien
OUT.IFC_GEOM(m)	Eine Map mit den Geometrieattributen wie IFC_GEOM, IFC_GEOM_LENGTH, etc.
OUT.IFC_GEOM(li of m)	Eine Liste von Maps mit den Geometrieattributen wie IFC_GEOM, IFC_GEOM_LENGTH, etc.

Mit einer Liste von Maps können unterschiedliche Geometrien in unterschiedlichen Ausprägungen für ein Objekt geschrieben werden. Zum Beispiel bei einer Leitung: die

Achse als Linien in einer bestimmten Farbe, das Profil als Körper in einer anderen Farbe.

Listenelemente für `OUT.IFC_PROPERTYSETS`

Beinhaltet eine Liste vom Maps mit den Propertysets.

Jede Map eines Propertysets beinhaltet einen Namen und eine Map mit den einzelnen Properties als Dupel Name/Wert.

Komponente	req/opt	Beschreibung
<code><m>.IFC_NAME(s)</code>	r	Name des Propertysets.
<code><m>.IFC_PROPERTY_SINGLEVALUES(m)</code>	r	Map mit den Property singlevalues des Propertysets. Jedes Property singlevalue mit Name und Wert in der Map.

11.4. Exportierte Prozeduren und Methoden

Prozedur `IFCOUT_OPEN ! [s output][[]]`
Beschreibung Öffnet das IFC-File `<output>` und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel `'c:\temp\test.ifc' IFCOUT_OPEN`

Prozedur `IFCOUT_WRITE_OBJECT0 ! [][[]]`
Beschreibung Schreibt ein Objekt in das Outputfile. Das OUT-Objekt muss gemäss dem Objektmodell vorbereitet sein. Die id des Objektes wird in `IFCOUT_VAR.IFC_OBJECT_ID` abgelegt..
Beispiel

```
&OUT MAPCLEAR

'IFCBUILDINGELEMENTPROXY' => OUT.IFC_CLASS
IN.Geometrie                => OUT.IFC_GEOM
'255/0/0'                   => OUT.IFC_COLOR

IFCOUT_WRITE_OBJECT

DISPLAY IFCOUT_VAR.IFC_OBJECT_ID
```

Prozedur `IFCOUT_CLOSE ! [][[]]`
Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel `IFCOUT_CLOSE`

Prozedur `IFCOUT_SET_DIMENSION2 ! r length, r width`
Beschreibung Setzt `OUT.IFC_GEOM_LENGTH`, `OUT.IFC_GEOM_WIDTH`
Beispiel Anwendung mit `SET_PARAM`.

```
2.0 SET_PARAM
1.5 SET_PARAM
IFCOUT_SET_DIMENSION2
```

Anwendung in einer Rule.

```
MAP INOUT
:
I1,<topic>,<table> => ...,IFCOUT_SET_DIMENSION2,2.0,1.5,...
:
END_MAP
```

Weitere Methoden mit analoger Anwendung:

Prozedur IFCOUT_SET_HEIGHT1 ! r height
Prozedur IFCOUT_SET_ROTATION1 ! r rotation
Prozedur IFCOUT_SET_GEOMETRY1 ! g geometry
Prozedur IFCOUT_SET_SCALE3 ! r xscale, r yscale, z zscale
Prozedur IFCOUT_SET_OFFSET3 ! r xoffset, r yoffset, z zoffset
Prozedur IFCOUT_SET_COLOR1 ! s rgb-color
Prozedur IFCOUT_SET_PROFILE1 ! s profile
Prozedur IFCOUT_SET_IFCBUILDINGELEMENTPROXYTYPE1 ! s ifc-file

Weitere Methoden zur Anwendung in Scripts.

Prozedur IFCOUT_WRITE_CLASS ! [m *o] [i id]
Beschreibung Schreibt ein Objekt. Das Objekt muss entsprechend dem Objektmodell im o-Objekt gesetzt werden. Die id des Objektes wird zurückgeliefert.
Beispiel &OUT IFCOUT_WRITE_CLASS => VAR.IFC_ID

Prozedur IFCOUT_WRITE_IFCRELAGGREGATES ! [s RelatingObject, s RelatedObjects] []
Beschreibung Schreibt ein Objekt der Klasse IFCRELAGGREGATES für die Beziehung von Objekten.
Beispiel '15' '16,17,18' IFCOUT_WRITE_IFCRELAGGREGATES

Prozedur IFCOUT_WRITE_IFCPROPERTYSET ! [s RelatedObjects] []
Beschreibung Schreibt zu Objekte Propertysets der Klasse IFCPROPERTYSET mit Values der Klasse IFCPROPERTYSINGLEVALUE. Die Propertysets müssen gemäss dem Objektmodell im OUT-Objekt gesetzt werden.
Beispiel '15' IFCOUT_WRITE_IFCPROPERTYSET

11.5. Skriptbeispiel

Beispiel ohne Datenbankbindung.

```
! Diese ICS Konfiguration kopiert einige von il2in.mod
! gelesenen Objekte mit odaifcout.mod in eine ifc-Datei.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE
  MESSAGE => 'Enter INTERLIS Input File'
  FILE_FILTER => xtf
  FILE_EXISTS => TRUE
```

```
OPT => input
END_MAP

MAP USER_INPUT2
  DIALOG => FILE
  MESSAGE => 'Enter IFC Output File'
  FILE_FILTER => ifc
  FILE_EXISTS => FALSE
  OPT => output
END_MAP

MAP IL2IN_PARAM
  LOG_TABLE      => ON
  TRACE          => OFF
  STATISTICS     => ON
  VALUE_CHECK   => OFF
  MODELS        => XTF
  MODEL_NAME    => OFF
  FORCE_COMPILE  => OFF
END_MAP

MAP IFCOUT_PARAM
  STATISTICS    => ON
END_MAP

MAP INPUT_SOURCES
  I1 => IL2IN,OPT.input
END_MAP

MAP INOUT
  I1          => IN.CLASS
  I1,LkPunkt  => P,CIRCLE,D,1.0,1.0,H,1.0,G,IN.SymbolPos,IFCOUT_WRITE_OBJECT0
  I1,LkLinie  => D,1.0,1.0,H,1.0,G,IN.Linie,IFCOUT_WRITE_OBJECT0
  I1,LkFlaeche => H,1.0,G,IN.Flaeche,IFCOUT_WRITE_OBJECT0
  I1,*       => OFF
END_MAP

MAP MACRO
  P => IFCOUT_SET_PROFILE1
  D => IFCOUT_SET_DIMENSION2
  H => IFCOUT_SET_HEIGHT1
  G => IFCOUT_SET_GEOMETRY1
END_MAP

| INCL \script\il2in.mod
| INCL \script\odaifcout.mod
| INCL \script\run1.prg
```

12. Modul IL2OUT - INTERLIS 2 schreiben

12.1. Allgemeines

Mit dem Modul IL2OUT können Objekte in eine INTERLIS 2 .xtf Datei geschrieben werden. Der Modul interpretiert neben der .xtf Datei auch die zugehörigen INTERLIS 2 Datenmodelle (.ili Dateien). Jedes Objekt wird von IL2OUT auf seine Konsistenz gegenüber den INTERLIS

2 Datenmodellen überprüft. Falls z.B. zwingende Attribute vergessen oder falsche Attributwerte gefunden werden, werden entsprechende Fehlermeldungen ausgegeben.

IL2OUT wird mit:

```
| INCL \script\il2out.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

12.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

12.3. Parametermap IL2OUT_PARAM

Folgende Parameter können in der Map IL2OUT_PARAM für den Modul IL2OUT gesetzt werden:

Parameter	req/opt	Beschreibung
FORCE_COMPILE	r	Falls dieser Parameter auf ON (Default = OFF) gesetzt wird, werden die INTERLIS Datenmodelle jedes mal mit dem INTERLIS 2.2 Compiler analysiert. Falls die Option auf OFF gesetzt ist, wird zuerst nachgeschaut, ob bereits eine vorcompilierte Version des Modells existiert (.ilo und .ilp Datei). Das Laden von vorcompilierten Modelldateien ist viel schneller möglich, daher ist die Option normalerweise auf OFF gesetzt.
MODEL_DIR	o	Normalerweise werden Modelldateien (.ili) zuerst in ICS_DIR\user\models2 gesucht. Mit der Option MODEL_DIR kann man einen alternativen (User-)Suchpfad angeben (Default = OFF).
MODELS	r	Modelldateien als kommaseparierte Liste (z.B. Units.ili, Time.ili, DM01AVCH24D.ili). Die Modelldateien müssen in der korrekten Reihenfolge für den INTERLIS 2 Compiler angegeben werden. Alternativ kann auch der Name einer Option angegeben werden, welche den Dateinamen einer .xtf Datei enthält (z.B. OPT.input).
ENCODER	o	Text nach UTF8 codieren (ON / OFF, Default = ON).
STATISTICS	o	Objektstatistik am Ende der .log Datei ausgeben (Default = OFF).
DEBUG	o	ON oder OFF. Debugmodus ein oder aus (Default = OFF).
XML_ID_CHECK	o	ON, OFF oder undefiniert. (Default undefiniert). INTERLIS TID's müssen einem XML Type ID entsprechen. Siehe dazu 3.3.1 Attribute Types > Validity constraint: ID">http://www.w3.org/TR/REC-xml > 3.3.1 Attribute Types > Validity constraint: ID . Ist dieser Parameter undefiniert, so wird immer das Zeichen x als Prefix dem Wert des TID vorangestellt, um sicherzustellen, dass die XML-Vorgabe erfüllt wird. Ist der Parameter mit ON definiert, wird geprüft, ob der TID bereits ein erlaubtes Zeichen am Anfang beinhaltet, nur wenn nicht wird ein x als Prefix dem Wert des TID vorangestellt. Ist der Parameter mit OFF definiert, wird der Wert des TID nicht geprüft und nicht verändert. Damit die INTERLIS Referenzen erhalten bleiben, gelten dieselben Regeln auch für die Werte der Referenzattribute.

12.4. Objektmodell

Der Modul IL2OUT verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.MODEL(s)	r	Model des OUT-Objekts.
OUT.TOPIC(s)	r	Topic des OUT-Objekt
OUT.CLASS(s)	r	Table des OUT-Objekts.
OUT.TID(s)	r	Transferidentifikation des OUT-Objekts .

Die restlichen Objektkomponenten sind abhängig von der dazugehörigen INTERLIS Klasse (s.a. OUT.MODEL, OUT.TOPIC bzw. OUT.CLASS). Alle INTERLIS Attribute werden als Komponenten des OUT-Objekts mit dem gleichem Namen erwartet. Die ICS Datentypen werden wie folgt auf INTERLIS Datentypen abgebildet:

INTERLIS Datentyp	ICS Datentyp
NUMBER	real oder int.
TEXT	string.
ENUMERATION	string.
STRUCTURE	map. Das XML-Tag der Struktur kann dem Label der Map entnommen werden (mit GET_LABEL).
LIST	list of map. Die XML-Tags der Strukturelemente können den Labeln der Maps entnommen werden.
BAG	list of map. Die XML-Tags der Strukturelemente können den Labeln der Maps entnommen werden.
ROLE	link.

12.5. Exportierte Prozeduren und Methoden

Prozedur	<code>IL2OUT_OPEN ! [s output][]</code>
Beschreibung	Erzeugt eine neue INTERLIS 2 Datei <output> und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'c:\test.xtf' IL2OUT_OPEN</code>
Prozedur	<code>IL2OUT_WRITE_OBJECT ! [] []</code>
Beschreibung	Schreibt das aktuelle OUT-Objekt in die geöffnete INTERLIS 2 Datei.
Beispiel	<code>IL2OUT_WRITE_OBJECT</code>
Prozedur	<code>IL2OUT_CREATE_LINK ! [s key,i orderpos][]</code>
Beschreibung	Erzeugt einen Link für das Schreiben einer Rolle. In der Outputdatei wird für die Rolle REF="<key>" und ORDERPOS="<orderpos>" gesetzt. ORDERPOS wird nur gesetzt, wenn <orderpos> > 0.
Beispiel	<code>'123' 0 IL2OUT_CREATE_LINK => OUT.Entstehung_von</code>
Prozedur	<code>IL2OUT_CLOSE ! [] []</code>

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel IL2OUT_CLOSE

12.6. Skriptbeispiel

```
! Diese ICS Konfiguration kopiert alle von IL2OUT
! gelesenen Objekte mit IL2OUT in in eine .xtf Datei.
```

```
|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .xtf Input File'
  FILE_FILTER => xtf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP USER_INPUT2
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .xtf Output File'
  FILE_FILTER => xtf
  FILE_EXISTS => FALSE
  OPT => output
END_MAP

MAP IL2IN_PARAM
  INPUT_EXTENSION => xtf
  MODELS           => XTF ! get from .xtf file
  FORCE_COMPILE    => OFF
  TRACE           => OFF
  STATISTICS      => ON
END_MAP

MAP IL2OUT_PARAM
  OUTPUT_EXTENSION => xtf
  MODELS           => OPT.input ! same as input
  FORCE_COMPILE    => OFF
  TRACE           => OFF
  STATISTICS      => ON
END_MAP

MAP INPUT_SOURCES
  I1 => IL2IN,OPT.input
END_MAP

MAP INOUT
  I1 => COPY_INOUT0
END_MAP

|INCL \script\util.lib
|INCL \script\il2in.mod
|INCL \script\il2out.mod
```

```

PROCEDURE COPY_INOUT0
  &IN &OUT MAPCOPY
  IL2OUT_WRITE_OBJECT
END_PROCEDURE

|INCL \script\run1.prg

```

13. Modul ILOUT - INTERLIS 1 schreiben

13.1. Allgemeines

Mit dem Modul ILOUT können Objekte in eine INTERLIS 1 .itf Datei geschrieben werden. Der Modul liest auch die zugehörigen INTERLIS 1 Datenmodelle (.ili Dateien). Jedes Objekt wird vom Modul auf seine Konsistenz gegenüber den INTERLIS 1 Datenmodellen überprüft. Falls z.B. zwingende Attribute vergessen oder falsche Attributwerte gefunden werden, werden entsprechende Fehlermeldungen ausgegeben.

Der Modul wird mit:

```
|INCL \script\ilout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

13.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

13.3. Parametermap ILOUT_PARAM

Folgende Parameter können in der Map ILOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
INTERLIS_DEF	r	Filename des INTERLIS Definitionsfiles. Die Angabe des Parameters ist obligatorisch
FORMAT	r	ITF oder XTF, Default = ITF. Definiert das Output Datenformat.
MATH_DEGREES	r	ON oder OFF, Default = OFF. DEGREES im mathematischen Sinn interpretieren, d.h. 0.0 = horizontal, Orientierung = Gegenuhrzeigersinn.
DOUBLEPOINT_CHECK	r	ON oder OFF, Default = OFF. Nacheinanderfolgende dopplete Punkte in Linien testen.
DEBUG	r	ON oder OFF, Default = OFF. Debugmodus ein oder aus.
STATUS	r	Enthält nach dem Schreiben eines Objektes einen Fehlerstatus, falls ein Fehler aufgetreten ist.
STATISTICS	r	ON oder OFF, Default = OFF. Statistik anzeigen.

13.4. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
------------	---------	--------------

OUT.TOPIC(s)	r	Topic des Objekts.
OUT.TABLE(s)	r	Table des Objekts.
OUT.OBJID(s)	r	Transferidentifikation des Objekts .

Die restlichen Objektkomponenten sind abhängig von der dazugehörigen INTERLIS Tabelle (s.a. OUT.TOPIC bzw. OUT.TABLE). Alle INTERLIS Attribute werden als Komponenten des OUT-Objekts mit dem gleichem Namen erwartet. Die ICS Datentypen werden wie folgt auf INTERLIS Datentypen abgebildet:

INTERLIS Daten- typ	ICS Datentyp
IRANGE	int
RRANGE	real
Text	string
GRADS	real
DEGREES	real
RADIANS	real
ENUMERATION	int oder string
COORD2	point
COORD3	point
POLYLINE	line
SURFACE	area
AREA	point (Zentroid) und area (Fläche)

13.5. Exportierte Prozeduren und Methoden

Prozedur	<code>ILOUT_OPEN ! [s input][[]]</code>
Beschreibung	Öffnet eine neue INTERLIS 1 Datei <input> und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'test.itf' ILOUT_OPEN</code>
Prozedur	<code>ILOUT_WRITE_OBJECT ! [][[]]</code>
Beschreibung	Schreibt das nächste OUT-Objekt in die geöffnete INTERLIS 1 Datei.
Beispiel	<code>ILOUT_WRITE_OBJECT</code>
Prozedur	<code>ILOUT_WRITE_SURFACE ! [s Surface-Table][[]]</code>
Beschreibung	Schreibt zu einem Hauptobjekt die Surface in die geöffnete INTERLIS 1 Datei. Muss nach ILOUT_WRITE_OBJECT angewendet werden. In OUT.OBJID muss die Transferidentifikation des Hauptobjekts, in IN.GEOM die Surface-Fläche enthalten sein.
Beispiel	<code>'Einzelobjekte' => OUT.TOPIC 'Flaechenelement' => OUT.TABLE '1' => OUT.OBJID ILOUT_WRITE_OBJECT 'Flaechenelement_Geometrie' ILOUT_WRITE_SURFACE</code>

Prozedur	<code>ILOUT_WRITE_SURFACE_LINEATTR ! [s Surface-Table, s Line-Attribute-Name][[]]</code>
Beschreibung	Wie ILOUT_WRITE_SURFACE mit dem zusätzlichen Schreiben eines eventuell in der Geometrie enthaltenen Linienattributs.
Beispiel	<pre>'Einzelobjekte' => OUT.TOPIC 'Flaechenelement' => OUT.TABLE '1' => OUT.OBJID ILOUT_WRITE_OBJECT 'Flaechenelement_Geometrie' 'Linienart' ILOUT_WRITE_SURFACE_LINEATTR</pre>
Prozedur	<code>ILOUT_WRITE_COMMENT ! [s Comment][[]]</code>
Beschreibung	Schreibt eine Kommentarzeile in den SCNT Abschnitt der .itf Datei.
Beispiel	<pre>'dies ist ein Test' ILOUT_WRITE_COMMENT</pre>
Prozedur	<code>ILOUT_CLOSE ! [][[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>ILOUT_CLOSE</pre>

13.6. Skriptbeispiel

```
! Diese ICS Konfiguration kopiert alle von ILIN
! gelesenen Objekte mit ILOUT in in eine .itf Datei.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP USER_INPUT2
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .itf Output File'
  FILE_FILTER => itf
  FILE_EXISTS => FALSE
  OPT => output
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF      => \models\Grunddatensatz.ili
  STATISTICS        => ON
END_MAP

MAP ILOUT_PARAM
  INTERLIS_DEF      => \models\Grunddatensatz.ili
  STATISTICS        => ON
END_MAP
```

```

MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
END_MAP

MAP INOUT
  I1 => COPY_INOUT0
END_MAP

|INCL \script\ilin.mod
|INCL \script\ilout.mod

PROCEDURE COPY_INOUT0
  &OUT MAPCLEAR
  &IN MAPRESET
  WHILE &IN MAPSCAN DO
    => VAR.ATTR
    => VAR.VALUE
    IF &VAR.VALUE GET_TYPE = 'ilink' THEN
      ! ilink in string umwandeln
      &VAR.VALUE ILIN.GET_ILINK_KEY => VAR.VALUE
    END_IF
    &OUT VAR.ATTR VAR.VALUE MAPINS
  END_WHILE
  ILOUT_WRITE_OBJECT
END_PROCEDURE

|INCL \script\run1.prg

```

14. Modul KMLOUT - Google KML schreiben

14.1. Allgemeines

Mit dem Modul KMLOUT können Objekte in eine KML Datei geschrieben werden.

Der Modul KMLOUT wird mit:

```
|INCL \script\kmlout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

14.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

14.3. Parametermap KMLOUT_PARAM

Folgende Parameter können in der Map KMLOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
TEMPLATE	r	Name der KML Templatedatei. Für die Templatedatei muss ein KML-File gemäss KML Spezifikationen angegeben werden. Das KML Templatefile beinhaltet alle Definitionen, auf die die zu

		schreibenden Objekte Bezug nehmen, wie zum Beispiel Style-Definitionen.
STATISTICS	r	ON oder OFF, Default = OFF. Objektstatistik am Ender der .log Datei anzeigen.
STROKE_TOL	r	<real> . Stroke-Tolerance zur Auflösung von Kreisbögen in Liniensegmente. Als Stroke-Tolerance können dieselben Werte wie für die ICS-Methode ICS.STROKE verwendet werden (s.a. iG/Script Benutzer- und Referenzhandbuch). Mit OFF werden die Kreisbögen nicht aufgelöst.

14.4. Objektmodell

Der Modul verlangt für jedes OUT-Objekt folgende Systemkomponenten:

Pro Objekttyp müssen ausserdem folgende Komponenten des OUT-Objekts gesetzt werden:

Allgemeine Komponenten für jedes OUT-Objekt

Komponente	req/opt	Beschreibung
OUT.FOLDER(s)	o	KML Folder - Ordner - in das das Objekt geschrieben wird. Handelt es sich um einen Unterordner, so ist der Unterordner mit dem gesamten Ordnerpfad anzugeben, z.B. Bodenbedeckung.Gebäude.Nummer.
OUT.GEOM(g)	o	Geometrie des Objektes.
OUT.<Tag>(s)	o	KML Tag. Definition weiterer KML Tags mit Werten für das Objekt. Die Tags und Werte richten sich nach dem Objekt und den KML Spezifikationen für das Objekt, z.B. styleUrl für eine Style-Definition des Objektes. Der Wert für das Tag kann auch eine ICS-Komponente sein, die den Wert beinhaltet. Ein Tag für ein Unterelement des Objektes muss der Tag-Pfad des Tags definiert werden, z.B. Polygon.extrude.
OUT.ExtendedData.<Tag>(s)	o	KML ExtendedData Tags. Definition weiterer Attribute mit Werten für das Objekt. Beispiel: ExtendedData.Identifikator => IN.Identifikator

Dieses Objektmodell gilt für die Anwendung mit der Prozedur KMLOUT_WRITE_OBJECT0. Das OUT-Objekt muss mit Prozeduren aufbereitet werden.

14.5. Record Definitionen

Mit Record Definitionen können vereinfacht Objekte definiert werden. Die Record Definitionen werden von der Prozedur KMLOUT_WRITE_RECORD1 verarbeitet. Eine Record Definition sieht wie folgt aus.

```
MAP <name>
  FOLDER => <Folder-Name>
  GEOM => <Geometrie-Komponente>
  :
  <Tag> => <Value|Komponente mit Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<name>

Eine Record Definition ist eine Map. Der Map-Name kann ein beliebiger Name sein.

FOLDER

Siehe Objektmodell.

<Tag>

Siehe Objektmodell.

Beispiel einer Record Definition.

```
MAP RECORD_POLYGON
  name           => Gebäude
  styleUrl       => #polygonstyle
  Polygon.extrude => 1
  Polygon.tessellate => 1
  Polygon.altitudeMode => relativeToGround
  GEOM           => IN.GEOM
  FOLDER         => Bodenbedeckung.Gebäude.Fläche
END_MAP
```

14.6. Folder Definitionen

Mit den Folder Definitionen können für die in den Record Definitionen verwendeten Folders KML Tags definiert werden. Die Folders Definition sieht wie folgt aus.

MAP FOLDERS

<Folder-Path>.<Tag> => <Value>

:

END_MAP

Die einzelnen Bestandteile einer Record-Definition sind:

FOLDERS

Die Folder Definition ist eine Map. Der Map-Name ist fix FOLDERS .

<Folder>-Path

Definition des Foldernames. Handelt es sich um einen Unterordner, muss der ganze Path für den Folder definiert werden. Folders in einem Path werden durch einen Punkt separiert.

Mit document für den Folder-Pfad können Tags für das Dokument definiert werden.

<Tag>

Tag für den Folder entsprechend den KML Spezifikationen.

<Value>

Wert für den Tag des Folders.

Zusätzlich zu den in der KML Spezifikation definierten Tags können folgende Modul-spezifischen Tags definiert werden.

<Tag>=ORDER

Tag für den Folder zum sortieren der Daten.

<Value>=ASCENDING|DESCENDING[, NUMERIC]

Definiert wie die Daten sortiert werden sollen. ASCENDING: aufwärts, DESCENDING: abwärts. Zusätzlich kann mit NUMERIC definiert werden, ob die Werte als numerische Werte interpretiert werden sollen und nicht als textuelle Werte. Macht nur Sinn, wenn alle Werte numerisch sind.

Beispiel einer Folder Definition.

```

MAP FOLDERS
  document.open                               => 1
  Bodenbedeckung.open                         => 1
  Bodenbedeckung.visibility                   => 1
  Bodenbedeckung.Gebäude.open                 => 1
  Bodenbedeckung.Gebäude.visibility           => 1
  Bodenbedeckung.Gebäude.Nummer.open         => 0
  Bodenbedeckung.Gebäude.Nummer.visibility   => 1
  Bodenbedeckung.Gebäude.Nummer.ORDER       => ASCENDING,NUMERIC
  Bodenbedeckung.Gebäude.Fläche.open         => 0
  Bodenbedeckung.Gebäude.Fläche.visibility   => 1
END_MAP

```

14.7. KML Templates

Mit dem Parameter `KMLOUT_PARAM.TEMPLATE` muss ein KML-Template definiert werden. Das KML Templatefile beinhaltet alle Definitionen, auf die die zu schreibenden Objekte Bezug nehmen, wie zum Beispiel Style-Definitionen. Nachfolgend ein Beispiel.

```

<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Document>
  <name>template.kml</name>

  <!-- Begin Style Definitions -->

  <Style id="polygonstyle_normal">
    <PolyStyle>
      <color>ff0000aa</color>
    </PolyStyle>
  </Style>
  <Style id="polygonstyle_highlite">
    <PolyStyle>
      <color>ff0000ff</color>
    </PolyStyle>
  </Style>
  <StyleMap id="polygonstyle">
    <Pair>
      <key>normal</key>
      <styleUrl>#polygonstyle_normal</styleUrl>
    </Pair>
    <Pair>
      <key>highlight</key>
      <styleUrl>#polygonstyle_highlite</styleUrl>
    </Pair>
  </StyleMap>

  <!-- End Style Definitions -->

</Document>
</kml>

```

14.8. Exportierte Prozeduren und Methoden

Prozeduren und Methoden

Prozedur	KMLOUT_OPEN ! [s file][[]
Beschreibung	Öffnet den KMLOUT Modul auf der Outputdei <file>. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>'c:\test.kml' KMLOUT_OPEN</pre>
Prozedur	KMLOUT_WRITE_OBJECT0
Beschreibung	Schreibt aus aufbereitete Objekt OUT in die aktuelle Outputdatei.
Beispiel	<pre>... => KMLOUT_WRITE_OBJECT0</pre>
Prozedur	KMLOUT_WRITE_RECORD1 ! record
Beschreibung	Schreibt das in record definierte Objekt in die aktuelle Outputdatei.
Beispiel	<pre>... => KMLOUT_WRITE_RECORD1,RECORD_PLOYGON</pre>
Prozedur	KMLOUT_CLOSE ! [[]]
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>KMLOUT_CLOSE</pre>
Triggers Folder	
Prozedur	KMLOUT_FOLDER_PRE_OPEN ! [[]]
Beschreibung	Trigger der aufgerufen wird, bevor ein Folder geöffnet wird. Der Folder steht in VAR.KML_FOLDER zur Verfügung.
Beispiel	<pre>PROCEDURE KMLOUT_FOLDER_PRE_OPEN ! code END_PROCEDURE</pre>
Prozedur	KMLOUT_FOLDER_POST_OPEN ! [[]]
Beschreibung	Trigger der aufgerufen wird, nachdem ein Folder geöffnet wurde. Der Folder steht in VAR.KML_FOLDER zur Verfügung.
Beispiel	<pre>PROCEDURE KMLOUT_FOLDER_POST_OPEN IF VAR.KML_FOLDER = 'Bodenbedeckung.Gebäude' THEN '<Region>' KMLOUT_WRITE_LINE KMLOUT_TAB_INC '<LatLonAltBox>' KMLOUT_WRITE_LINE ! box '</LatLonAltBox>' KMLOUT_WRITE_LINE KMLOUT_TAB_DEC '</Region>' KMLOUT_WRITE_LINE END_IF END_PROCEDURE</pre>
Prozedur	KMLOUT_FOLDER_PRE_CLOSE ! [[]]
Beschreibung	Trigger der aufgerufen wird, bevor ein Folder geschlossen wird. Der Folder steht in VAR.KML_FOLDER zur Verfügung.

Beispiel

```
PROCEDURE KMLOUT_FOLDER_PRE_CLOSE
! code
END_PROCEDURE
```

Prozedur

KMLOUT_FOLDER_POST_CLOSE ! [[]]

Beschreibung

Trigger der aufgerufen wird, nachdem ein Folder geschlossen wurde. Der Folder steht in VAR.KML_FOLDER zur Verfügung.

Beispiel

```
PROCEDURE KMLOUT_FOLDER_POST_CLOSE
! code
END_PROCEDURE
```

Triggers Placemark (Objects)**Prozedur**

KMLOUT_PLACEMARK_PRE_OPEN ! [[]]

Beschreibung

Trigger der aufgerufen wird, bevor ein Placemark geöffnet wird. Der Placemark steht in OUT zur Verfügung.

Beispiel

```
PROCEDURE KMLOUT_PLACEMARK_PRE_OPEN
! code
END_PROCEDURE
```

Prozedur

KMLOUT_PLACEMARK_POST_OPEN ! [[]]

Beschreibung

Trigger der aufgerufen wird, nachdem ein Placemark geöffnet wurde. Der Placemark steht in OUT zur Verfügung.

Beispiel

```
PROCEDURE KMLOUT_PLACEMARK_POST_OPEN
! code
END_PROCEDURE
```

Prozedur

KMLOUT_PLACEMARK_PRE_CLOSE ! [[]]

Beschreibung

Trigger der aufgerufen wird, bevor ein PLACEMARK geschlossen wird. Der Placemark steht in OUT zur Verfügung.

Beispiel

```
PROCEDURE KMLOUT_PLACEMARK_PRE_CLOSE
! code
END_PROCEDURE
```

Prozedur

KMLOUT_PLACEMARK_POST_CLOSE ! [[]]

Beschreibung

Trigger der aufgerufen wird, nachdem ein Placemark geschlossen wurde. Der Placemark steht in OUT zur Verfügung.

Beispiel

```
PROCEDURE KMLOUT_PLACEMARK_POST_CLOSE
! code
END_PROCEDURE
```

Weitere Prozeduren**Prozedur**

KMLOUT_WRITE_LINE ! [s string][[]]

Beschreibung

Eine zusätzliche Zeile in das Output-File schreiben. Siehe Beispiel in KMLOUT_FOLDER_POST_OPEN.

Beispiel

```
'<Region>' KMLOUT_WRITE_LINE
```

Prozedur

KMLOUT_TAB_INC ! [[]]

Beschreibung Den Tabulator für das Schreiben in das Output-File um einen Einschub erhöhen. Siehe Beispiel in KMLOUT_FOLDER_POST_OPEN.

Beispiel KMLOUT_TAB_INC

Prozedur KMLOUT_TAB_DEC ! [][]

Beschreibung Den Tabulator für das Schreiben in das Output-File um einen Einschub verkleinern. Siehe Beispiel in KMLOUT_FOLDER_POST_OPEN.

Beispiel KMLOUT_TAB_DEC

14.9. Skriptbeispiel

```
! Diese ICS Konfiguration transferiert aus INTERLIS
! gelesenen Objekte mit KMLOUT in in eine .kml Datei.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG => FILE
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP
```

```
MAP USER_INPUT2
  DIALOG => FILE
  MESSAGE => 'Enter .kml Output File'
  FILE_FILTER => kml
  FILE_EXISTS => FALSE
  OPT => output
END_MAP
```

```
MAP ILIN_PARAM
  INTERLIS_DEF => \models\dm01avch24d.ili
  LOG_TABLE => ON
  TRACE => OFF
  STATISTICS => ON
  CALC_SURFACE => ON
  ENUM_TO_TEXT => ON
END_MAP
```

```
MAP ILIN_TOPO
  Bodenbedeckung,BoFlaeche_Geometrie => AREA
  DEFAULT => OFF
END_MAP
```

```
MAP KMLOUT_PARAM
  TEMPLATE => \data\template.kml
  STATISTICS => ON
  STROKE_TOL => 0.001
  FENCE_MODE => OVERLAP
END_MAP
```

```
MAP FOLDERS
```

```
Bodenbedeckung.Gebäude.Nummer.open      => 0
Bodenbedeckung.Gebäude.Nummer.visibility => 1
Bodenbedeckung.Gebäude.Fläche.open       => 0
Bodenbedeckung.Gebäude.Fläche.visibility => 1
END_MAP

MAP RECORD_POINT
name           => IN.GebaeudenummerPos_von.Nummer
styleUrl       => #pointstyle
visibility     => 1
Point.extrude  => 1
Point.tessellate => 1
Point.altitudeMode => relativeToGround
GEOM           => IN.Pos
FOLDER         => Bodenbedeckung.Gebäude.Nummer
ExtendedData.Nummer => IN.GebaeudenummerPos_von.Nummer
ExtendedData.GWR_EGID => IN.GebaeudenummerPos_von.GWR_EGID
END_MAP

MAP RECORD_POLYGON
name           => Gebäude
styleUrl       => #polygonstyle
Polygon.extrude => 1
Polygon.tessellate => 1
Polygon.altitudeMode => relativeToGround
GEOM           => IN.GEOM
FOLDER         => Bodenbedeckung.Gebäude.Fläche
END_MAP

MAP INPUT_SOURCES
I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
I1 => IN.TOPIC,IN.TABLE
I1,Bodenbedeckung,BoFlaeche_Area      => IN.Art
I1,Bodenbedeckung,BoFlaeche_Area,0    => R_1,RECORD_POLYGON
I1,Bodenbedeckung,GebaeudenummerPos   => R_1,RECORD_POINT
I1,* => OFF
END_MAP

MAP MACRO
R_1 => KMLOUT_WRITE_RECORD1
END_MAP

| INCL \script\iltopo.mod
| INCL \script\kmlout.mod
| INCL \script\run1.prg
```

15. Modul MYSQLOUT - MySQL-Datenbank schreiben

15.1. Allgemeines

Mit dem Skriptmodul können Objekte in eine MySQL-Datenbank via ODBC geschrieben werden. Der Modul unterstützt speziell die MySQL Extension Spatial für räumliche Daten. Der Modul kann mit RUN1 verwendet werden.

MYSQLOUT unterstützt die Geometry-Typen POINT, LINESTRING, POLYGON von MySQL nach der OGC Simple Feature Specification.

Der Modul beinhaltet folgende Besonderheiten:

- Schreibt Daten nach MySQL.
- Legt Tabellen für die Daten an.
- Schreibt die notwendigen Definitionen für MySQL.
- Schreibt die Geometrien für MySQL nach OGC Simple Feature Specification.
- Schreibt den Spatial Index für die Geometrien von MySQL.
- Die Datenbank kann nach dem Schreiben der Daten direkt mit MySQL weiterbearbeitet werden.

Der Modul wird mit:

```
|INCL \script\mysqlout.mod
```

in einer ICS Konfiguration verfügbar gemacht.

15.2. Abhängigkeiten von anderen Modulen

Der Modul MYSQLOUT ist eine Erweiterung des Moduls DBOUT. Alle im Modul DBOUT beschriebenen Anteile gelten auch für das Modul MYSQLOUT. Ziehen Sie deshalb die Dokumentation des Modules DBOUT bei, insbesondere die Abschnitte über die Parametermaps DB_PARAM und DBOUT_PARAM.

15.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.

TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.
-------	---	--

15.4. Parametermap DBOUT_PARAM

Folgende Parameter können in der Map DBOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
CREATE_TABLE	o	ON, OFF, Default = OFF. Definiert ob die Tables definiert mit Record-Definitionen (siehe weiter unten) in der Datenbank erzeugt werden sollen. Mit OFF werden die Tabellen nicht erzeugt. Mit ON werden die Tabellen erzeugt und zwar zum Zeitpunkt, wenn das erste Objekt in die Tabelle geschrieben wird. Tables in die keine Objekte geschrieben werden, werden auch nicht erzeugt.
DELETE_OLD	o	ON, OFF oder DROP, Default = OFF. Löscht bestehende Daten in Tables definiert mit Record-Definitionen (siehe weiter unten). Mit OFF werden keine Daten gelöscht. Mit ON werden die Daten über ein delete-sql-statement gelöscht. Mit DROP wird die Table und damit die Daten gelöscht. Zusammen mit DROP und dem Parameter CREATE_TABLE => ON werden die Tables gelöscht und wieder erzeugt.
SQLTRACE	o	ON oder OFF, Default = OFF. Zeigt alle sql-statements im Logfile an.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltet werden. Sie dazu mehr unter dem Kapitel Datasets. Der Wert ON hebt den Parameter DELETE_OLD auf.

15.5. Parametermap MYSQLOUT_PARAM

Folgende Parameter können in der Map MYSQLOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SPATIAL_STROKE	o	<real> , Default = 0.001. PostGIS basiert auf der OGC Simple Feature Specification. Diese Spezifikation unterstützt keine Kreisbögen, deshalb müssen Kreisbögen in Liniensegmente aufgelöst werden. Als Stroke-Tolerance können dieselben Werte wie für die ICS-Methode ICS.STROKE verwendet werden (s.a. iG/Script Benutzer- und Referenzhandbuch).
SPATIAL_INDEXCREATE	o	ON oder OFF, Default = OFF. Definiert ob die Indexe für die Geometrie-Spalten für MySQL erzeugt werden sollen. Aufgrund der Tabelle und des Attributes wird automatisch ein Indexname erzeugt.
SPATIAL_INDEXDROP	o	ON oder OFF, Default = OFF. Definiert ob die Indexe für die Geometrie-Spalten vor dem Schreiben der Daten gelöscht werden sollen. Dies ermöglicht das schnellere Schreiben der Daten. Mit MYSQLOUT_PARAM.SPATIAL_INDEXCREATE => ON werden die Indizes nach dem Schreiben wieder angelegt.
SPATIAL_SRID	o	<integer> oder OFF, Default = OFF. Definiert die MySQL-SRID-Identifikation für die Geometrien. Jede Geometrie wird mit dem

		definierten SRID nach MySQL geschrieben. Das SRID muss in MySQL definiert sein.
--	--	---

 MySQL erlaubt das Indexieren von Spatial Typen nur, wenn der Geometry-Type mit NOT NULL definiert ist, und daher auch keine NULL Einträge vorkommen. Die zu transferierenden Daten garantieren diese NOT NULL Definition in der Regel nicht. Deshalb sollte unter MySQL ein Geometry Attribut nicht als NOT NULL definiert werden und auch der Spatial Index nicht automatisch generiert werden. Die Indexierung sollte nachträglich analysiert und vorgenommen werden.

15.6. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tablename des OUT Objekts.
OUT.<Attribut>(s)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Type des Wertes muss mit dem Type des Attributes in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden. Geometrien werden mit der MySQL-Funktion <code>GeomFromText(OGC-WKT-Geometry)</code> geschrieben. Eine ICS-Geometry wird in eine OGC-WKT-Geometry mit der Methode <code>OGC.GEOM2WKT</code> umgewandelt (siehe w. u.).

Dieses Objektmodell gilt für die Anwendung mit der Prozedur `MYSQLOUT_WRITE_OBJECT0`. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfiguration mit Prozeduren und Methoden vorgängig kreiert werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, so beachten Sie das Kapitel mit den Record-Definitionen und die Prozedur `MYSQLOUT_WRITE_RECORD1`.

15.7. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank angelegt werden. Die Record Definitionen werden von der Prozedur `MYSQLOUT_WRITE_RECORD1` verarbeitet. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<n>
  TABLE => <Table-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix `RECORD_` beginnen und eine eindeutige Nummer `<n>` für die Record Definition beinhalten.

TABLE

Diese Komponente ist obligatorisch und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente `TABLE` die Datenbank-Tabelle.

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen innerhalb der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

CHAR(<length>)

Stringtypen sind immer als CHAR mit der Länge <length> zu definieren. Je nach Datenbank wird der Typ vom Modul in einen der Datenbank bekannten Type umgewandelt z.B. MySQL als VARCHAR.

NUMBER

Number-Typen ohne Argumente werden vom Modul als NUMBER(38,5) interpretiert.

OGC_GEOMETRY(<type>;<dimension>)

PostGIS Spatial Geometrien müssen als Type OGC_GEOMETRY definiert werden. Dabei sind folgende weiteren Definitionen notwendig.

<type>

Type der Geometrie, einer der Werte: point|line|area.

<dimension>

Dimension der Geometrie, einer der Werte: 2D|3D.



MySQL 5.0 unterstütz 3D nicht. Deshalb ignoriert der Modul 3D und verwendet 2D.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können Konstanten oder ICS-Variablen, die einen Wert beinhalten - z.B. IN.OBJID - verwendet werden.

Beispiel einer Record Definition.

```
MAP RECORD_1
TABLE => Fixpunkte_LFP
OBJID => CHAR(10),IN.OBJID
ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
NUMMER => CHAR(12),IN.Nummer
GEOMETRIE => OGC_GEOMETRY(point;2D),IN.Geometrie
LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
BEGEHBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
SYMBOLORI => NUMBER,IN.SymbolOri
ART_TXT => CHAR(4),IN.Art_TXT
HERKUNFT => CHAR(30),IN.Herkunft
END_MAP
```

15.8. Datasets

Zur Verwaltung von Datasets in der Datenbank ist das analoge Kapitel des Modules DBOUT zu beachten.

15.9. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MODEL

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell inklusive den notwendigen MySQL Spatial Definitionen erstellt werden soll, so ist der Parameter CONFIG_PARAM.GENERATE_MODEL auf ON zu setzen und das Script `il2mysql.lib` zu includen:

```
MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
:
|INCL \script\il2db\il2pgres.lib
```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell angelegt, falls es nicht schon angelegt wurde.

15.10. Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie der Modul DBOUT. Ziehen Sie deshalb die Dokumentation des Modules DBOUT bei.

Zusätzlich stellt der Modul MYSQLOUT folgende Prozeduren und Methoden zur Verfügung.

Prozedur	<code>MYSQLOUT_OPEN [] []</code>
Beschreibung	Öffnet eine Datenbank definiert mit <code>DB_PARAM.SOURCE</code> . Die Prozedur wird von <code>RUN1</code> automatisch aufgerufen.
Beispiel	<code>MYSQLOUT_OPEN</code>
Prozedur	<code>MYSQLOUT_WRITE_OBJECT0</code>
Beschreibung	Schreibt einen Objekt in die Datenbank. Das OUT-Objekt muss gemäss dem Objektmodell vorbereitet sein.
Beispiel	<code>MYSQLOUT_WRITE_OBJECT0</code>
Prozedur	<code>MYSQLOUT_WRITE_RECORD1 ! s recordname</code>
Beschreibung	Schreibt ein Objekt definiert in <code><recordname></code> in die Datenbank. <code><recordname></code> ist der Name einer Record-Definition (siehe weiter oben). Je nach den gesetzten Parametern führt die Prozedur beim ersten Aufruf für eine Record-Definition folgende Aktionen auf der Datenbank aus: <ol style="list-style-type: none"> 1. Die Tabelle oder deren Inhalt wird gelöscht, falls definiert durch Parameter. 2. Die Tabelle wird erzeugt, falls definiert durch Parameter. 3. Die MySQL Metadefinitionen werden generiert. 4. Das Objekt wird in die Datenbank geschrieben.
Beispiel	<code>... => MYSQLOUT_WRITE_RECORD1, RECORD_1</code>
Prozedur	<code>MYSQLOUT_CLOSE [] []</code>

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel `MYSQLOUT_CLOSE`

Methode `OGC.GEOM2WKT [g|li geometry][s ogc-wkt-geometry]`

Beschreibung Übersetzt eine ICS-Geometrie point,line oder area in eine OGC WKT Geometrie als String. Als Input können auch Listen von Geometrien übergeben werden. Falls die Geometrien nicht übersetzt werden können, wird auf dem Stack ein NULL-String zurückgegeben. (WKT: Well Known Text nach OGC Simple Feature Specification).

Beispiel `IN.GEOM OGC.GEOM2WKT => VAR.GEOM`

Folgende Konversionen werden durchgeführt:

point

to OGC-POINT

line

to OGC-LINESTRING

area

to OGC-POLYGON

list of points

to OGC-MULTIPOINT

list of lines

to OGC-MULTILINESTRING

list of areas

to OGC-MULTIPOLYGON

list of points and/or lines and/or areas

to OGC-GEOMETRYCOLLECTION

Neben diesen Prozeduren des Modules stehen auch die Prozeduren und Methoden des Modules DBOUT zur Verfügung. Diese Prozeduren und Methoden sind im Modul DBOUT beschrieben.

15.11. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,
! kreiert eine Tabelle für LFP's in der Datenbank
! und schreibt die LFP's in die Tabelle.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter INTERLIS Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT         => input
END_MAP
```

```
MAP USER_INPUT2
  DIALOG      => ODBC
  OPT         => output
```

```

END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\Grunddatensatz.ili
  STATISTICS   => ON
  CALC_SURFACE => ON
  ENUM_TO_TEXT => ON
  TRACE        => OFF
END_MAP

MAP ILIN_TOPO
  DEFAULT => OFF
END_MAP

MAP DB_PARAM
  SOURCE   => '' ! ODBC-Source
  USER     => '' ! ODBC-User
  PASSWD   => '' ! ODBC-Password
  TRACE    => OFF
END_MAP

MAP DBOUT_PARAM
  STATISTICS   => ON
  CREATE_TABLE => ON
  DATASET      => ON
END_MAP

MAP MYSQLOUT_PARAM
  SPATIAL_STROKE      => 0.001 ! a real Stroke-Tolerance
  SPATIAL_INDEXDROP  => ON   ! ON|OFF Spatial Index Drop   before Insert
  SPATIAL_INDEXCREATE => ON   ! ON|OFF Spatial Index Create after Insert
  SPATIAL_SRID       => -1
END_MAP

MAP RECORD_1
  TABLE => Fi_LFP
  OBJID  => CHAR(10),IN.OBJID
  ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
  NUMMER   => CHAR(12),IN.Nummer
  GEOMETRIE => OGC_GEOMETRY(point;3D),IN.Geometrie
  LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
  HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
  BEGEHBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
  SYMBOLORI => NUMBER,IN.SymbolOri
  ART_TXT => CHAR(4),IN.Art_TXT
  HERKUNFT => CHAR(30),IN.Herkunft
END_MAP

MAP INPUT_SOURCES
  I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
  I1          => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => MYSQLOUT_WRITE_RECORD1,RECORD_1
  I1,*       => OFF
END_MAP

```

```

| INCL \script\iltopo.mod
| INCL \script\mysqlout.mod
| INCL \script\run1.prg

```

16. Modul ORAOUT - Oracle-Datenbank schreiben

16.1. Allgemeines

Mit dem Skriptmodul können Objekte in eine Oracle-Datenbank via ODBC geschrieben werden. Der Modul unterstützt speziell die Oracle Spatial Option für räumliche Daten. Der Modul kann mit RUN1 verwendet werden.

ORAOUT unterstützt sämtliche Geometry-Typen von Oracle Spatial 9.2 und 10g. Diese sind POINT, LINE, POLYGON, MULTIPOINT, MULTILINE, MULTIPOLYGON und COLLECTION.

Der Modul beinhaltet folgende Besonderheiten:

- Schreibt Daten nach Oracle Spatial.
- Legt Tabellen für die Daten an.
- Schreibt die notwendigen Definitionen für Oracle Spatial.
- Schreibt die Geometrien für Oracle Spatial.
- Schreibt den Spatial Index für die Geometrien von Oracle Spatial.
- Die Datenbank kann nach dem Schreiben der Daten direkt mit Oracle Spatial weiterbearbeitet werden.

Der Modul wird mit:

```

| INCL \script\oraout.mod

```

in einer ICS Konfiguration verfügbar gemacht.

16.2. Abhängigkeiten von anderen Modulen

Der Modul ORAOUT ist eine Erweiterung des Moduls DBOUT. Alle im Modul DBOUT beschriebenen Anteile gelten auch für das Modul ORAOUT. Ziehen Sie deshalb die Dokumentation des Modules DBOUT bei, insbesondere die Abschnitte über die Parametermaps DB_PARAM und DBOUT_PARAM.

16.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.

PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

16.4. Parametermap DBOUT_PARAM

Folgende Parameter können in der Map DBOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
CREATE_TABLE	o	ON, OFF, Default = OFF. Definiert ob die Tables definiert mit Record-Definitionen (siehe weiter unten) in der Datenbank erzeugt werden sollen. Mit OFF werden die Tabellen nicht erzeugt. Mit ON werden die Tabellen erzeugt und zwar zum Zeitpunkt, wenn das erste Objekt in die Tabelle geschrieben wird. Tables in die keine Objekte geschrieben werden, werden auch nicht erzeugt.
DELETE_OLD	o	ON, OFF oder DROP, Default = OFF. Löscht bestehende Daten in Tables definiert mit Record-Definitionen (siehe weiter unten). Mit OFF werden keine Daten gelöscht. Mit ON werden die Daten über ein delete-sql-statement gelöscht. Mit DROP wird die Table und damit die Daten gelöscht. Zusammen mit DROP und dem Parameter CREATE_TABLE => ON werden die Tables gelöscht und wieder erzeugt.
BATCH	o	ON oder OFF, Default = OFF. Mit OFF werden die sql-statements direkt auf der Datenbank ausgeführt. Mit ON werden die sql-statements in ein Batchfile geschrieben. Mit dem Parameter BATCH_FILE wird das Batchfile definiert.
BATCH_OUTPUT_DIR	o	<directory>. Definiert ein Output-Directory für Batchfiles. Mit diesem Parameter kann das Output-Directory für Batchfiles definiert werden, falls der Parameter BATCH = ON definiert ist. Batchfiles können sein ein File mit SQL-Statements oder in Kombination mit dem Oracle Output Modul die SQLLOADER-Bulkfiles. Ist dieser Parameter nicht gesetzt, so wird das Output-Directory aus einem eventuellen Input-File definiert in OPT.input bestimmt. Ist kein Input-File definiert, so ist das Output-Directory iltools\data\ics.sql.
BATCH_FILE	o	<file>. Definiert das Batchfile. Mit diesem Parameter kann das Batchfile definiert werden, falls der Parameter BATCH = ON definiert ist. Ist dieser Parameter nicht gesetzt, so wird das Batchfile aus einem eventuellen Input-File definiert in OPT.input mit der Endung .sql bestimmt. Ist kein Input-File definiert, so ist das Batchfile iltools\data\ics.sql als definiert. Das Batchfile beinhaltet SQL-Statements, um die transferierten Daten mittels SQL in eine Datenbank zu importieren.
SQLTRACE	o	ON oder OFF, Default = OFF. Zeigt alle sql-statements im Logfile an.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltet werden. Sie dazu mehr unter dem Kapitel Datasets. Der Wert ON hebt den Parameter DELETE_OLD auf.

16.5. Parametermap ORAOUT_PARAM

Folgende Parameter können in der Map ORAOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SQLLOADER_USE	o	ON oder OFF, Default = OFF. Definiert ob das Schreiben der Daten in die Datenbank über die Oracle-Utility SQLLoader erfolgen soll. SQLLoader ermöglicht das schnelle Laden grosser Datenmengen in Oracle Spatial. Ist dieser Parameter auf ON gesetzt, so werden sogenannte Bulk-Files für SQLLoader erstellt. Ist der Parameter DBOUT_PARAM.BATCH auf OFF gesetzt, so werden die Bulk-Files erstellt und mit SQLLoader während des Transfers in die Datenbank gelesen. Ist der Parameter DBOUT_PARAM.BATCH auf ON gesetzt, so werden lediglich die Bulk-Files erstellt. Diese Variante ermöglicht das Erstellen von Bulk-Files zur späteren Weiterverarbeitung oder zur Abgabe an Dritte.
SQLLOADER_CMD	o	<command>. Default undefiniert. Ist der Parameter ORAOUT_PARAM.SQLLOADER_USE auf ON gesetzt muss dieser Parameter mit dem Command für die Utility SQLLoader gesetzt sein. Das zu setzende Command entspricht dem Befehl, wie er auf einer Commandline für SQLLoader angewendet werden muss. Beispiel: sqlldr.exe userid=scott/tiger@ORACL. Anstatt des Commands kann im Parameter auch der Verweis auf eine Datei definiert werden, welche das Command beinhaltet.
SQLLOADER_DECIMAL-POINT	o	<char>. Default , . Definiert den Dezimalpunkt für reelle Zahlen in den SQLLoader-Bulk-Files.
SQLLOADER_CONTI-NUECODE	o	<string>. Default # . Definiert die Fortsetzungszeichen in den SQLLoader-Bulk-Files.
SQLLOADER_FIELDSE-PARATOR	o	<string>. Default . Definiert die Spaltentrennzeichen in den SQLLoader-Bulk-Files.
SPATIAL_STROKE	o	<real> oder OFF, Default = OFF. Stroke-Tolerance zur Auflösung von Kreisbögen in Liniensegmente. Als Stroke-Tolerance können dieselben Werte wie für die ICS-Methode ICS.STROKE verwendet werden (s.a. iG/Script Benutzer- und Referenzhandbuch). Mit OFF werden die Kreisbögen nicht aufgelöst.
SPATIAL_METAIN-SERT	o	ON oder OFF, Default = ON. Definiert ob die Metadaten für Oracle Spatial in die Tabelle user_sdo_geom_metadata geschrieben werden sollen.
SPATIAL_INDEXCREA-TE	o	ON oder OFF, Default = ON. Definiert ob die Indexe für die Geometrie-Spalten für Oracle Spatial erzeugt werden sollen. Aufgrund der Tabelle und des Attributes wird automatisch ein Indexname erzeugt.
SPATIAL_INDEXDROP	o	ON oder OFF, Default = ON. Definiert ob die Indexe für die Geometrie-Spalten vor dem Schreiben der Daten gelöscht werden sollen. Dies ermöglicht das schnellere Schreiben der Daten. Mit ORAOUT_PARAM.SPATIAL_INDEXCREATE => ON werden die Indizes nach dem Schreiben wieder angelegt.
SPATIAL_INDEXTYPE	o	QTREE oder RTREE, Default = RTREE. Definiert den Indextyp für die Geometrie-Spalten von Oracle Spatial.
SPATIAL_INDEXTA-BLESPACE	o	<tablespace> oder OFF, Default = OFF. Definiert den Tablespace für die Indizes der Geometrie-Spalten von Oracle Spatial.

SPATIAL_VALIDATE	o	ON oder OFF, Default = OFF. Definiert ob Oracle Spatial SQL-Statements in das Logfile gesschrieben werden sollen. Mit diesen Statements können nachträglich unter Oracle Spatial die Geometrien validiert werden.
SPATIAL_SRID	o	<integer> oder OFF, Default = OFF. Definiert die ORacle-SRID-Identifikation für die Geometrien. Jede Geometrie wird mit dem definierten SRID nach Oracle geschrieben.
SPATIAL_META_X	o	<min,max,resolution>. Default 0,1000000,0.001. Definiert für die Metadaten von Oracle Spatial die Ausdehnung und Auflösung der X-Koordinaten.
SPATIAL_META_Y	o	<min,max,resolution>. Default 0,1000000,0.001. Definiert für die Metadaten von Oracle Spatial die Ausdehnung und Auflösung der Y-Koordinaten.
SPATIAL_META_Z	o	<min,max,resolution>. Default 0,1000000,0.001. Definiert für die Metadaten von Oracle Spatial die Ausdehnung und Auflösung der Z-Koordinaten.
SPATI-AL_GEO_CLEAN	o	ON oder OFF, Default =OFF . Definiert ob die Geometrien für Oracle aufbereitet weden sollen. Mit OFF werden die Geometrien nicht aufbereitet. Mit ON werden die Geometrien aufbereitet. Mehr zur Funktion GEOM_CLEAN siehe im Benutzerhandbuch iG/Script im Appendix.

Für die Anwendung der Oracle Spatial Option und der Oracle-Utility SQLLoader ist die entsprechende Dokumentation von Oracle zu beachten.

16.6. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tablename des OUT Objekts.
OUT.<Attribut>(s)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Type des Wertes muss mit dem Type des Attributes in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden.

Dieses Objektmodell gilt für die Anwendung mit der Prozedur ORAOUT_WRITE_OBJECT0. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfiguration mit Prozeduren und Methoden vorgängig kreiert werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, so beachten Sie das Kapitel mit den Record-Definitionen und die Prozedur ORAOUT_WRITE_RECORD1.

16.7. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank angelegt werden. Die Record Definitionen werden von der Prozedur ORAOUT_WRITE_RECORD1 verarbeitet. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<n>
  TABLE => <Table-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
```

:
END_MAP

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix `RECORD_` beginnen und eine eindeutige Nummer `<n>` für die Record Definition beinhalten.

TABLE

Diese Komponente ist obligatorisch und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente `TABLE` die Datenbank-Tabelle.

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen innerhalb der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

CHAR(<length>)

Stringtypen sind immer als `CHAR` mit der Länge `<length>` zu definieren. Je nach Datenbank wird der Typ vom Modul in einen der Datenbank bekannten Type umgewandelt z.B. Oracle als `VARCHAR2`.

NUMBER

Number-Typen ohne Argumente werden vom Modul als `NUMBER(38,5)` interpretiert.

DATE("YYYY-MM-DD")

Datums Typ. Mit dem Datums Typ muss auch das Format des Datums definiert werden. Der Modul erzeugt aufgrund des Formats eine entsprechende SQL-Anweisung `TO_DATE('1993-04-03', 'YYYY-MM-DD')`. Der Wert für das Attribut, muss als Integer oder String die Form `YYYYMMDD` aufweisen. Zum Beispiel `<Attribute-Value>=19939493` oder `<Attribute-Value>='19939493'`.

MDSYS.SDO_GEOMETRY(<type>;<dimension>;<resolution>)

Oracle Spatial Geometrien müssen als Type `MDSYS.SDO_GEOMETRY` definiert werden. Dabei sind folgende weiteren Definitionen notwendig.

<type>

Type der Geometrie, einer der Werte: `point|line|area|gmtext`.

<dimension>

Dimension der Geometrie, einer der Werte: `2D|3D|3DM`.

`3DM` nur für Geometriotyp `line` erlaubt. Die 3 Koordinate wird als Measurement interpretiert.

<resolution>

Real-Wert der Auflösung.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können Konstanten oder ICS-Variablen, die einen Wert beinhalten - z.B. `IN.OBJID` - verwendet werden.

Beispiel einer Record Definition.

```

MAP RECORD_1
  TABLE => Fixpunkte_LFP
  OBJID => CHAR(10),IN.OBJID
  ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
  NUMMER => CHAR(12),IN.Nummer
  GEOMETRIE => MDSYS.SDO_GEOMETRY(point;3D;0.001),IN.Geometrie
  LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
  HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
  BEGEBBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
  SYMBOLORI => NUMBER,IN.SymbolOri
  ART_TXT => CHAR(4),IN.Art_TXT
  HERKUNFT => CHAR(30),IN.Herkunft
END_MAP

```

16.8. Datasets

Zur Verwaltung von Datasets in der Datenbank ist das analoge Kapitel des Modules DBOUT zu beachten.

16.9. Datenbank Modellgenerierung mit CONFIG_PARAM.GENERATE_MODEL

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell inklusive den notwendigen Oracle Spatial Definitionen erstellt werden soll, so ist der Parameter CONFIG_PARAM.GENERATE_MODEL auf ON zu setzen und das Script il2ora.lib zu includen:

```

MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
:
|INCL \script\il2db\il2ora.lib

```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell angelegt, falls es nicht schon angelegt wurde.

16.10. Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie der Modul DBOUT. Ziehen Sie deshalb die Dokumentation des Modules DBOUT bei.

Zusätzlich stellt der Modul ORAOUT folgende Prozeduren und Methoden zur Verfügung.

Prozedur	<code>ORAOUT_OPEN [] []</code>
Beschreibung	Öffnet eine Datenbank definiert mit <code>DB_PARAM.SOURCE</code> . Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>ORAOUT_OPEN</code>
Prozedur	<code>ORAOUT_WRITE_OBJECT0</code>
Beschreibung	Schreibt einen Objekt in die Datenbank. Das OUT-Objekt muss gemäss dem Objektmodell vorbereitet sein.
Beispiel	<code>ORAOUT_WRITE_OBJECT0</code>

Prozedur	<code>ORAOUT_WRITE_RECORD1 ! s recordname</code>
Beschreibung	<p>Schreibt ein Objekt definiert in <recordname> in die Datenbank. <recordname> ist der Name einer Record-Definition (siehe weiter oben). Je nach den gesetzten Parametern führt die Prozedur beim ersten Aufruf für eine Record-Definition folgende Aktionen auf der Datenbank aus:</p> <ol style="list-style-type: none"> 1. Die Tabelle oder deren Inhalt wird gelöscht, falls definiert durch Parameter. 2. Die Tabelle wird erzeugt, falls definiert durch Parameter. 3. Die Oracle Spatial Metadefinitionen werden generiert, falls definiert durch Parameter. 4. Das Objekt wird in die Datenbank geschrieben.
Beispiel	<code>... => ORAOUT_WRITE_RECORD1,RECORD_1</code>
Prozedur	<code>ORAOUT_CLOSE [][[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>ORAOUT_CLOSE</code>
Methode	<code>ORACLE.GEOM_TO_SDO_GEOMETRY [g li geom, i dimension [, b measurement]][s sdo-geometry]</code>
Beschreibung	Übersetzt eine ICS-Geometrie point,line oder area in eine Oracle-Spatial Geometrie als String. Als Input können auch Listen von Geometrien übergeben werden. Falls die Geometrien nicht übersetzt werden können, wird auf dem Stack ein NULL-String zurückgegeben. Boolean measurement ist optional und nur für dimension 3 und Geometrien vom Type line erlaubt. Die 3 Koordinate wird als Measurement interpretiert.
Beispiel	<code>IN.GEOM 2 ORACLE.GEOM_TO_SDO_GEOMETRY => VAR.SDO_GEOM</code>
	<p>Folgende Konversionen werden durchgeführt:</p> <p>point to SDO-point</p> <p>line to SDO-line</p> <p>area to SDO-polygon</p> <p>list of points to SDO-multipoints</p> <p>list of lines to SDO-multilines</p> <p>list of areas to SDO-multipolygon</p> <p>list of points and/or lines and/or areas to SDO-collection</p>

Methode	<code>ORACLE.SET_RESOLUTION [r resolution][[]]</code>
Beschreibung	Definiert die Resolution mit der die Koordination von Geometrien in eine Oracle-Spatial Geometrie übersetzt werden. Default ist 0.001. Zum Beispiel wird mit der Resolution 0.00001 eine Koordinate mit 5 Nachkommastellen übersetzt.
Beispiel	<code>0.001 ORACLE.SET_RESOLUTION</code>

Neben diesen Prozeduren des Modules stehen auch die Prozeduren und Methoden des Modules `DBOUT` zur Verfügung. Diese Prozeduren und Methoden sind im Modul `DBOUT` beschrieben.

16.11. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,
! kreiert eine Tabelle für LFP's in der Datenbank
! und schreibt die LFP's in die Tabelle.
```

```
|LICENSE \license\iltoolspro.lic

MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter INTERLIS Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT         => input
END_MAP

MAP USER_INPUT2
  DIALOG      => ODBC
  OPT         => output
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\Grunddatensatz.ili
  STATISTICS   => ON
  CALC_SURFACE => ON
  ENUM_TO_TEXT => ON
  TRACE        => OFF
END_MAP

MAP ILIN_TOPO
  DEFAULT     => OFF
END_MAP

MAP DB_PARAM
  SOURCE      => '' ! ODBC-Source
  USER        => '' ! ODBC-User
  PASSWD      => '' ! ODBC-Password
  TRACE       => OFF
END_MAP

MAP DBOUT_PARAM
  STATISTICS   => ON
  CREATE_TABLE => ON
  DATASET      => ON
END_MAP
```

```

MAP ORAOUT_PARAM
  SPATIAL_STROKE          => OFF ! OFF or a real Stroke-Tolerance
  SPATIAL_METAINSERT     => ON  ! ON|OFF Spatial Meta Insert
  SPATIAL_INDEXDROP      => ON  ! ON|OFF Spatial Index Drop   before Insert
  SPATIAL_INDEXCREATE    => ON  ! ON|OFF Spatial Index Create after Insert
  SPATIAL_VALIDATE      => OFF ! ON|OFF Spatial Validate after Insert
  SPATIAL_META_X         => 0,1000000,0.001 ! x-min,x-max,x-tolerance
  SPATIAL_META_Y         => 0,1000000,0.001 ! y-min,y-max,y-tolerance
  SPATIAL_META_Z         => 0,1000000,0.001 ! z-min,z-max,z-tolerance
  SPATIAL_INDEXTABLESPACE => OFF
  SPATIAL_SRID           => OFF
END_MAP

MAP RECORD_1
  TABLE => Fi_LFP
  OBJID  => CHAR(10),IN.OBJID
  ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
  NUMMER  => CHAR(12),IN.Nummer
  GEOMETRIE => MDSYS.SDO_GEOMETRY(point;3D;0.001),IN.Geometrie
  LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
  HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
  BEGEBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
  SYMBOLORI => NUMBER,IN.SymbolOri
  ART_TXT => CHAR(4),IN.Art_TXT
  HERKUNFT => CHAR(30),IN.Herkunft
END_MAP

MAP INPUT_SOURCES
  I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
  I1          => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => ORAOUT_WRITE_RECORD1,RECORD_1
  I1,*       => OFF
END_MAP

| INCL \script\iltopo.mod
| INCL \script\oraout.mod
| INCL \script\run1.prg

```

17. Modul PGRESOUT - PostGreSQL/PostGIS-Datenbank schreiben

17.1. Allgemeines

Mit dem Skriptmodul können Objekte in eine PostGreSQL-Datenbank via ODBC geschrieben werden. Der Modul unterstützt speziell die PostGreSQL Option PostGIS für räumliche Daten. Der Modul kann mit RUN1 verwendet werden.

PGRESOUT unterstützt sämtliche Geometry-Typen von PostGIS.

Der Modul beinhaltet folgende Besonderheiten:

- Schreibt Daten nach PostGreSQL/PostGIS.
- Legt Tabellen für die Daten an.
- Schreibt die notwendigen Definitionen für PostGreSQL/PostGIS.
- Schreibt die Geometrien für PostGIS nach OGC Simple Feature Specification.
- Schreibt den Spatial Index für die Geometrien von PostGIS.
- Die Datenbank kann nach dem Schreiben der Daten direkt mit PostGIS weiterbearbeitet werden.

Der Modul wird mit:

```
| INCL \script\pgresout.mod
```

in einer ICS Konfiguration verfügbar gemacht.

17.2. Abhängigkeiten von anderen Modulen

Der Modul PGRESOUT ist eine Erweiterung des Moduls DBOUT. Alle im Modul DBOUT beschriebenen Anteile gelten auch für das Modul PGRESOUT. Ziehen Sie deshalb die Dokumentation des Modules DBOUT bei, insbesondere die Abschnitte über die Parametermaps DB_PARAM und DBOUT_PARAM.

17.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
SOURCE	o	ODBC-Source der Datenbank. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
USER	o	Datenbank User für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
PASSWORD	o	Datenbank Password für die Verbindung mit der ODBC-Source. Muss gesetzt werden, falls die ODBC-Source nicht interaktiv abgefragt wird.
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

17.4. Parametermap DBOUT_PARAM

Folgende Parameter können in der Map DBOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
CREATE_TABLE	o	ON, OFF, Default = OFF. Definiert ob die Tables definiert mit Record-Definitionen (siehe weiter unten) in der Datenbank erzeugt werden sollen. Mit OFF werden die Tabellen nicht erzeugt. Mit ON werden die Tabellen erzeugt und zwar zum Zeitpunkt, wenn das erste Objekt in die Tabelle geschrieben wird. Tables in die keine Objekte geschrieben werden, werden auch nicht erzeugt.

DELETE_OLD	o	ON, OFF oder DROP, Default = OFF. Löscht bestehende Daten in Tables definiert mit Record-Definitionen (siehe weiter unten). Mit OFF werden keine Daten gelöscht. Mit ON werden die Daten über ein delete-sql-statement gelöscht. Mit DROP wird die Table und damit die Daten gelöscht. Zusammen mit DROP und dem Parameter CREATE_TABLE => ON werden die Tables gelöscht und wieder erzeugt.
SQLTRACE	o	ON oder OFF, Default = OFF. Zeigt alle sql-statements im Logfile an.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltet werden. Sie dazu mehr unter dem Kapitel Datasets. Der Wert ON hebt den Parameter DELETE_OLD auf.

17.5. Parametermap PGRESOUT_PARAM

Folgende Parameter können in der Map PGRESOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
WITH_OIDS	o	ON oder OFF, Default = undefiniert. Definiert ob Tabellen mit OID's kreiert werden sollen. Ist der Parameter auf ON gesetzt, werden die Tabellen mit OID's kreiert. Ist der Parameter auf OFF gesetzt, werden die Tabellen ohne OID's kreiert. Ist der Parameter nicht definiert, gilt die PostGreSQL Konfiguration Variable default_with_oids, welche definiert, ob Tabellen mit OID's kreiert werden sollen. OID ist ein PostGreSQL-Systemattribut und beinhaltet einen eindeutigen Objektschlüssel. Zum Beispiel der MapServer verwendet diese OID's.
SPATIAL_STROKE	o	<real> , Default = 0.001. PostGIS basiert auf der OGC Simple Feature Specification. Diese Spezifikation unterstützt keine Kreisbögen, deshalb müssen Kreisbögen in Liniensegmente aufgelöst werden. Als Stroke-Tolerance können dieselben Werte wie für die ICS-Methode ICS.STROKE verwendet werden (s.a. iG/Script Benutzer- und Referenzhandbuch).
SPATIAL_INDEXCREATE	o	ON oder OFF, Default = ON. Definiert ob die Indexe für die Geometrie-Spalten für PostGIS erzeugt werden sollen. Aufgrund der Tabelle und des Attributes wird automatisch ein Indexname erzeugt.
SPATIAL_INDEXDROP	o	ON oder OFF, Default = ON. Definiert ob die Indexe für die Geometrie-Spalten vor dem Schreiben der Daten gelöscht werden sollen. Dies ermöglicht das schnellere Schreiben der Daten. Mit PGRESOUT_PARAM.SPATIAL_INDEXCREATE => ON werden die Indizes nach dem Schreiben wieder angelegt.
SPATIAL_SRID	o	<integer> oder OFF, Default = OFF. Definiert die PostGIS-SRID-Identifikation für die Geometrien. Jede Geometrie wird mit dem definierten SRID nach PostGIS geschrieben. Das SRID muss in der PostGIS-Systemtabelle spatial_ref_sys definiert sein.

17.6. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tablename des OUT Objekts.
OUT.<Attribut>(s)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Type des Wertes muss mit dem Type des Attributes in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden. Geometrien werden mit der PostGIS-Funktion <code>GeometryFromText(OGC-WKT-Geometry)</code> geschrieben. Eine ICS-Geometry wird in eine OGC-WKT-Geometry mit der Methode <code>OGC.GEOM2WKT</code> umgewandelt (siehe w. u.).

Dieses Objektmodell gilt für die Anwendung mit der Prozedur `PGRESOUT_WRITE_OBJECT0`. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfiguration mit Prozeduren und Methoden vorgängig kreiert werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, so beachten Sie das Kapitel mit den Record-Definitionen und die Prozedur `PGRESOUT_WRITE_RECORD1`.

17.7. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank angelegt werden. Die Record Definitionen werden von der Prozedur `PGRESOUT_WRITE_RECORD1` verarbeitet. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<n>
  TABLE => <Table-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix `RECORD_` beginnen und eine eindeutige Nummer `<n>` für die Record Definition beinhalten.

TABLE

Diese Komponente ist obligatorisch und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente `TABLE` die Datenbank-Tabelle.

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen innerhalb der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

CHAR(<length>)

Stringtypen sind immer als `CHAR` mit der Länge `<length>` zu definieren. Je nach Datenbank wird der Typ vom Modul in einen der Datenbank bekannten Type umgewandelt z.B. `PostgreSQL` als `VARCHAR2`.

NUMBER

Number-Typen ohne Argumente werden vom Modul als `double precision` interpretiert.

INTEGER

Integer.

DATE

Date.

OGC_GEOMETRY(<type>;<dimension>)

PostGIS Spatial Geometrien müssen als Type `OGC_GEOMETRY` definiert werden. Dabei sind folgende weiteren Definitionen notwendig.

<type>

Type der Geometrie, einer der Werte: `point|line|area`.

<dimension>

Dimension der Geometrie, einer der Werte: `2D|3D`.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können Konstanten oder ICS-Variablen, die einen Wert beinhalten - z.B. `IN.OBJID` - verwendet werden.

Beispiel einer Record Definition.

```
MAP RECORD_1
  TABLE => Fixpunkte_LFP
  OBJID => CHAR(10),IN.OBJID
  ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
  NUMMER => CHAR(12),IN.Nummer
  GEOMETRIE => OGC_GEOMETRY(point;3D),IN.Geometrie
  LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
  HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
  BEGEHBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
  SYMBOLORI => NUMBER,IN.SymbolOri
  ART_TXT => CHAR(4),IN.Art_TXT
  HERKUNFT => CHAR(30),IN.Herkunft
END_MAP
```

17.8. Datasets

Zur Verwaltung von Datasets in der Datenbank ist das analoge Kapitel des Modules `DBOUT` zu beachten.

17.9. Datenbank Modellgenerierung mit `CONFIG_PARAM.GENERATE_MODEL`

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell inklusive den notwendigen PostGreSQL/PostGIS Spatial Definitionen erstellt werden soll, so ist der Parameter `CONFIG_PARAM.GENERATE_MODEL` auf `ON` zu setzen und das Script `il2pgres.lib` zu includen:

```
MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
:
|INCL \script\il2db\il2pgres.lib
```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell angelegt, falls es nicht schon angelegt wurde.

17.10. Prozeduren und Methoden

Der Modul beinhaltet alle Prozeduren und Methoden wie der Modul DBOUT. Ziehen Sie deshalb die Dokumentation des Modules DBOUT bei.

Zusätzlich stellt der Modul PGRESOUT folgende Prozeduren und Methoden zur Verfügung.

Prozedur `PGRESOUT_OPEN [][]`
Beschreibung Öffnet eine Datenbank definiert mit `DB_PARAM.SOURCE`. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel `PGRESOUT_OPEN`

Prozedur `PGRESOUT_WRITE_OBJECT0`
Beschreibung Schreibt einen Objekt in die Datenbank. Das OUT-Objekt muss gemäss dem Objektmodell vorbereitet sein.
Beispiel `PGRESOUT_WRITE_OBJECT0`

Prozedur `PGRESOUT_WRITE_RECORD1 ! s recordname`
Beschreibung Schreibt ein Objekt definiert in `<recordname>` in die Datenbank. `<recordname>` ist der Name einer Record-Definition (siehe weiter oben). Je nach den gesetzten Parametern führt die Prozedur beim ersten Aufruf für eine Record-Definition folgende Aktionen auf der Datenbank aus:

1. Die Tabelle oder deren Inhalt wird gelöscht, falls definiert durch Parameter.
2. Die Tabelle wird erzeugt, falls definiert durch Parameter.
3. Die PostGIS Metadefinitionen werden generiert.
4. Das Objekt wird in die Datenbank geschrieben.

Beispiel `... => PGRESOUT_WRITE_RECORD1,RECORD_1`

Prozedur `PGRESOUT_CLOSE [][]`
Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel `PGRESOUT_CLOSE`

Methode `OGC.GEOM2WKT [g|li geometry][s ogc-wkt-geometry]`
Beschreibung Übersetzt eine ICS-Geometrie point,line oder area in eine OGC WKT Geometrie als String. Als Input können auch Listen von Geometrien übergeben werden. Falls die Geometrien nicht übersetzt werden können, wird auf dem Stack ein NULL-String zurückgegeben. (WKT: Well Known Text nach OGC Simple Feature Specification).

Beispiel `IN.GEOM OGC.GEOM2WKT => VAR.GEOM`

Folgende Konversionen werden durchgeführt:

point
to OGC-POINT

line
to OGC-LINESTRING

area
to OGC-POLYGON

list of points
to OGC-MULTIPOINT

list of lines
to OGC-MULTILINESTRING

list of areas
to OGC-MULTIPOLYGON

list of points and/or lines and/or areas
to OGC-GEOMETRYCOLLECTION

Neben diesen Prozeduren des Modules stehen auch die Prozeduren und Methoden des Modules DBOUT zur Verfügung. Diese Prozeduren und Methoden sind im Modul DBOUT beschrieben.

17.11. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,  
! kreiert eine Tabelle für LFP's in der Datenbank  
! und schreibt die LFP's in die Tabelle.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1  
  DIALOG      => FILE  
  MESSAGE     => 'Enter INTERLIS Input File'  
  FILE_FILTER => itf  
  FILE_EXISTS => TRUE  
  OPT         => input  
END_MAP
```

```
MAP USER_INPUT2  
  DIALOG      => ODBC  
  OPT         => output  
END_MAP
```

```
MAP ILIN_PARAM  
  INTERLIS_DEF => \models\Grunddatensatz.ili  
  STATISTICS   => ON  
  CALC_SURFACE => ON  
  ENUM_TO_TEXT => ON  
  TRACE        => OFF  
END_MAP
```

```
MAP ILIN_TOPO  
  DEFAULT => OFF  
END_MAP
```

```
MAP DB_PARAM
```

```

SOURCE      => '' ! ODBC-Source
USER        => '' ! ODBC-User
PASSWD      => '' ! ODBC-Password
TRACE       => OFF
END_MAP

MAP DBOUT_PARAM
STATISTICS  => ON
CREATE_TABLE => ON
DATASET     => ON
END_MAP

MAP PGRESOUT_PARAM
SPATIAL_STROKE      => 0.001 ! a real Stroke-Tolerance
SPATIAL_INDEXDROP   => ON   ! ON|OFF Spatial Index Drop   before Insert
SPATIAL_INDEXCREATE => ON   ! ON|OFF Spatial Index Create after Insert
SPATIAL_SRID        => -1
END_MAP

MAP RECORD_1
TABLE => Fi_LFP
OBJID => CHAR(10),IN.OBJID
ENTSTEHUNG => CHAR(10),IN.Entstehung.OBJID
NUMMER => CHAR(12),IN.Nummer
GEOMETRIE => OGC_GEOMETRY(point;3D),IN.Geometrie
LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
BEGEHBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
SYMBOLORI => NUMBER,IN.SymbolOri
ART_TXT => CHAR(4),IN.Art_TXT
HERKUNFT => CHAR(30),IN.Herkunft
END_MAP

MAP INPUT_SOURCES
I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
I1          => IN.TOPIC,IN.TABLE
I1,Fixpunkte,LFP => PGRESOUT_WRITE_RECORD1,RECORD_1
I1,*       => OFF
END_MAP

|INCL \script\iltopo.mod
|INCL \script\pgresout.mod
|INCL \script\run1.prg

```

18. Modul PSOUT - PostScript (PDF, JPEG, TIF) schreiben

18.1. Allgemeines

Mit dem Modul PSOUT können Objekte in eine PostScript Datei geschrieben werden. Mit Zusatzsoftware - kommerzielle, Freeware, Shareware - wie zum Beispiel GhostScript kann eine PostScript Datei in ein anderes Format wie PDF, JPEG, TIF, etc. transferiert werden.

Der Modul wird mit:

```
INCL \script\psout.mod
```

in einer ICS Konfiguration verfügbar gemacht.

 Der Modul ist *nicht* mit RUN1 kompatibel und muss daher zusammen mit der speziellen .prg Datei \script\il2ps\il2ps.prg verwendet werden.

18.2. Abhängigkeiten von anderen Modulen

Für Plot Layout Elemente wie Titelblatt, Koordinatenkreuze, etc. ist das Verarbeitungsmodul PLOT einzubeziehen. Mehr dazu in der Dokumentation des Modules PLOT.

18.3. Parametermap PSOUT_PARAM

Folgende Parameter können in der Map PSOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
FILE_FORMAT	o	ps, pdf, jpg, tif oder tif_tfw. Definiert das Dateiformat für den Output. Beim Typ tif_tfw wird zusätzlich zur TIF-Datei noch eine TFW-Datei erzeugt.
ORIGIN	o	<x>/<y>.Definiert der Ursprung in Koordinaten unten links der Daten für den Output. Bei der Verwendung des Modules PLOT.MOD wird dieser Parameter mit PLOT_PARAM.ORIGIN überschrieben.
ANGLE	o	<r>.Definiert den Drehwinkel der Daten für den Output. Bei der Verwendung des Modules PLOT.MOD wird dieser Parameter mit PLOT_PARAM.ANGLE überschrieben.
SCALE	o	1:<scale>, Default = 1:500. Definiert den Massstab für den Output. Bei der Verwendung des Modules PLOT.MOD wird dieser Parameter mit PLOT_PARAM.SCALE überschrieben.
WIDTH	o	<m> . Definiert die Breite im Massstab 1:1 der Daten in Meter für den Output. Bei der Verwendung des Modules PLOT.MOD wird dieser Parameter mit PLOT_PARAM.WIDTH überschrieben.
HEIGHT	o	<m> .Definiert die Höhe im Massstab 1:1 der Daten in Meter für den Output. Bei der Verwendung des Modules PLOT.MOD wird dieser Parameter mit PLOT_PARAM.HEIGHT überschrieben.
PIXELSIZE	o	<r>, Default = 0.001. Definiert die Pixelgröße eines Pixels für den Output. Ein positiver Wert definiert wieviele Meter der Daten im Massstab 1:1 einem Pixel entsprechen, z.B. eine Breite der Daten von 300 Metern und eine Pixelgröße von 0.1 Meter ergibt einen Output mit einer Breite von 3000 Pixel. Ein negativer Wert defi-

		niert als Faktor, wie gross ein Pixel im Verhältnis zur gesamten Breite des Outputs ist, z.B. -0.001 ergibt einen Output mit einer Breite von 1000 Pixel. Der Parameter wird bei der Verwendung des Modules PLOT.MOD für einen PLOT_PARAM.TYPE => RASTER berücksichtigt.
PDF_DATA_FORMAT	o	jpg oder OFF, Default OFF. Definiert das Daten Format innerhalb eines PDF's bei einem Output Format PDF. Das heisst, die Daten im PDF-File werden als JPG eingebettet.
PDF_DATA_RESOLUTION	o	<dpi>, Default = 300. Definiert die Auflösung der Daten als dots per inch innerhalb eines PDF's bei einem Output Format PDF, falls als Daten Format auf jpg gesetzt ist.
SYMBOLOLOGY[<n>]	o	<Symbology-Library>, Default = ". Mit diesem Parameter können eine oder mehrere Signatur-Bibliotheken definiert werden. Signatur-Bibliotheken sind INTERLIS Dateien die Signaturen entsprechend dem INTERLIS Modell Symbology.ili beinhalten. Mehr dazu unter dem Abschnitt zu Signaturen. Bei der Verwendung mehrerer Signatur-Bibliotheken muss man dem Parameternamen eine fortlaufende Nummer beginnend mit 1 anfügen.
STROKE_TOLERANCE	o	<r>, Default = 0.001. Definiert die Toleranz wie Kreisbögen in Liniensegmente aufgelöst werden sollen. Der Wert ist identisch mit dem Argument für die Methode ICS.STROKE.
GS_LIB	o	<Path>, Default = ". Für die Transformation der PostScript Files in ein anderes Format PDF,JPG,TIF,etc. ist der Modul vorbereitet, dies mit dem Shareware Produkt GhostScript der Firma Aladdin Enterprises auszuführen. Dazu müssen Sie das Produkt GhostScript vom Internet laden und installieren. Mit diesem Parameter definieren Sie das Installationsverzeichnis des Produktes GhostScript. Falls der Parameter nicht gesetzt ist, wird das Installationsverzeichnis von GhostScript noch nach folgender Reihenfolge gesucht: 1.Verzeichnis definiert mit der Umgebungsvariable GS_LIB. 2. Annahme GhostScript ist unter ILTOOLS_DIR\re-dist\gs_x86 installiert.
DEBUG	o	ON oder OFF, Default = OFF. Debugmodus ein oder aus.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.

18.4. Symbole Clippen

Mit der optionalen Map PSOUT_SYMBOL_CLIP können Freistellflächen für Symbole definiert werden. Diese Map beinhaltet folgende Definitionen:

```
MAP PSOUT_SYMBOL_CLIP
...
<symbol> => <ON|OFF>,[<color>],[<rand>],[<priority>],[<clipsymbol>],[<clipsymbolscale>]
...
DEFAULT => <ON|OFF>,[<color>],[<rand>],[<priority>],[<clipsymbol>],[<clipsymbolscale>]
END_MAP

<symbol>
```

Required. Definiert das Symbol für welches die Definition gilt. Mit dem Eintrag DEFAULT kann eine Definition für alle anderen Symbole definiert werden.

<ON|OFF>

Required. Definiert ob das Symbol geclipt werden soll oder nicht.

<color>

Optional. Definiert die Farbe für die Clip-Fläche. Default=white.

<rand>

Optional. Definiert einen zusätzlichen Rand für die Clip-Fläche. Default=0.0.

<priority>

Optional. Definiert eine Priorität für die Clip-Fläche. Default=identische Priorität wie die Priorität des <symbol>.

<clipsymbol>

Optional. Definiert ein Symbol für die Clip-Fläche. Default=kein Clipsymbol. Die Clip-Fläche wird aus dem Symbol bestimmt. Soll ein anderes Symbol als Clip-Fläche verwendet werden, so ist hier das entsprechende Symbol einzutragen..

<clipsymbolscale>

Optional. Definiert eine Skalierung für die Clipp-Fläche, die aus dem <clipsymbol> bestimmt wird. Default=identische Skalierung wie die Skalierung des <symbol>. Diese Skalierung skaliert die Skalierung des <symbol>. Diese Skalierung ist keine absolute Skalierung, sondern eine relative Skalierung des bereits skalierten Symbols <symbol>

Beispiel einer Definition:

```
MAP PSOUT_SYMBOL_CLIP
! <symbol> => <ON|OFF>,[<color>],[<rand>],[<priority>],[<clipsymbol>],[<clipsymbolscale>]
097055      => OFF ! Koordinatenkreuz
DEFAULT    => ON
END_MAP
```

18.5. Signaturen

Signaturen (Symbole, Farben, etc.) können als Signatur-Bibliotheken von INTERLIS Dateien im Modell ILTOOLS_DIR\system\symb\Symbology.ili verwendet werden. Im demselben Directory sind diverse Beispiel solcher Signatur-Bibliotheken. Um Signatur-Bibliotheken zu erstellen, gibt es diverse Wege. In der Regel sind Signaturen wie zum Beispiel Symbole bereits in einem CAD-System vorhanden. Mit einer Konfiguration CAD-System zu INTERLIS können solche Symbole nach INTERLIS transferiert werden.

18.6. True Type Fonts

Einleitung

Sie können bei den Textelementen True Type Fonts verwenden.

Installation

Falls der True Type Font noch nicht in Windows installiert ist, installieren Sie den Font mit

Windows > Systemsteuerung > Schriftarten > Datei > Neue Schriftwart Installieren ...

Zum Beispiel die Cadastra Schriftarten, die unter

ICS_DIR\system\font

Ca.ttf

CadastraBd.ttf

CadastraIt.ttf

CaBI.ttf

CadastraSymbol.ttf

CadastraMask.ttf

abgelegt sind.

Namen

Bei der Verwendung der True Type Fonts in einer Konfiguration müssen Sie die Namen der Fonts wie folgt verwenden.

Beispiel Font:

Font Arial	
True Type Font Name	ICS Konfiguration Name
Arial	Arial
Arial Bold	Arial-Bold
Arial Italic	Arial-Italic
Arial Italic Bold	Arial-BoldItalic

Speziell beim Cadastra Font folgende Namen verwenden:

Font Cadastra	
True Type Font Name	ICS Konfiguration Name
Cadastra	Cadastra-Roman
Cadastra Bold	Cadastra-Bold
Cadastra Italic	Cadastra-Italic
Cadastra Italic Bold	Cadastra-BoldItalic
CadastraSymbol	CadastraSymbol
CadastraSymbol Mask	CadastraSymbol-Mask

Mask

Fonts können maskiert werden. Verwenden Sie im Fontnamen den Suffix Masked.

Beispiel:

Arial-BoldMasked

Verwendung

Der Font wird wie folgt angewendet:

```
... => PSOUT_WRITE_TEXT12,IN.Objekt.Name,IN.Pos, IN.Ori,IN.HAli,IN.VAli,Arial-Bold,2,0,black,,0,0,0.06,4
```

18.7. GhostScript

Für die Transformation der PostScript Files in ein anderes Format PDF,JPG,TIF,etc. ist der Modul vorbereitet, dies mit dem Shareware Produkt GhostScript der Firma Aladdin Enterprises auszuführen. Dazu müssen Sie das Produkt GhostScript vom Internet laden und installieren. Mit dem PSOUT_PARAM.GS_LIB definieren Sie das Installationsverzeichnis des Produktes

GhostScript. Falls der Parameter nicht gesetzt ist, wird das Installationsverzeichnis von GhostScript noch nach folgender Reihenfolge gesucht: 1. Verzeichnis definiert mit der Umgebungsvariable GS_LIB. 2. Annahme GhostScript ist unter ILTOOLS_DIR\redist\gs_x86 installiert.

18.8. Objektmodell

Allgemeine Komponenten für jedes OUT-Objekt

Komponente	req/opt	Beschreibung
OUT.TYPE(s)	r	Typ des Objekts.
OUT.PRIORITY(i)	r	Priorität des Objekts. Ein Objekt mit der Priorität n wird über ein Objekt mit der Priorität n-1 gezeichnet.
OUT.COLOR(i)	o	Farbe des Objekts. Integer als RGB Werte in der Form $(R*256*256) + (G*256) + B$ wobei R,G oder B einen Wert von 0 bis 255 aufweist.

Zusätzliche Komponenten für OUT.TYPE = 'POLYLINE'

Komponente	req/opt	Beschreibung
OUT.GEOM(l)	r	Linien-Geometrie des Objekts.
OUT.LSTYLE(s)	o	Linestyle des Objektes. Unterstützt werden continuous, dotted, dashed, dash.dot, dash.dot.dot, dot.dash.dash.
OUT.WIDTH(r)	o	Breite des Objekts.

Zusätzliche Komponenten für OUT.TYPE = 'POLYGON'

Komponente	req/opt	Beschreibung
OUT.GEOM(a)	r	Flächen-Geometrie des Objekts.

Zusätzliche Komponenten für OUT.TYPE = 'TEXT'

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Punkt-Geometrie des Objekts.
OUT.TXT(s)	r	Text des Objekts.
OUT.FONT(s)	r	Font des Objekts. z.B. Helvetica, Helvetica-Oblique, Helvetica-Bold, Helvetica-BoldOblique, etc. Mit dem Suffix Masked zum Font kann definiert werden, dass der Font "maskiert" (d.h. freigestellt) werden soll.
OUT.ROT(r)	o	Orientierung des Objekts.
OUT.HEIGHT(r)	o	Höhe des Objekts.
OUT.HEIGHT(r)	o	Höhe des Objekts.
OUT.XSCALE(r)	o	Horizontale Streckung des Objekts.
OUT.SLANT(r)	o	Slant des Objekts.
OUT.HALI(i)	o	Horizontales Alignment des Objekts.
OUT.VALI(i)	o	Vertikales Alignment des Objekts.

Zusätzliche Komponenten für OUT.TYPE = 'CLIP'

Komponente	req/opt	Beschreibung
------------	---------	--------------

OUT.GEOM(a)	r	Flächen-Geometrie des Objekts. In der Regel ist nur eine Clip-Fläche zu schreiben. Die Clip-Fläche definiert, welches Gebiet der Daten dargestellt werden soll. Die Clip-Fläche ist nur wirksam für Objekte, die eine höhere Priorität n+i als die Clip-Fläche n haben.
-------------	---	---

18.9. Exportierte Prozeduren und Methoden

- Prozedur** PSOUT_OPEN ! [s input][]
- Beschreibung** Öffnet eine neue PostScript Datei <input> und initialisiert den Modul. Die Prozedur wird on RUN1 automatisch aufgerufen.
- Beispiel** `'test.ps' PSOUT_OPEN`
- Prozedur** PSOUT_CLOSE ! [[]]
- Beschreibung** Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
- Beispiel** `PSOUT_CLOSE`
- Prozedur** PSOUT_WRITE_POLYLINE5 ! l geometrie, s lstyle, s color, r width, i priority
- Beschreibung** Schreibt eine Linien-Geometrie. Die Prozedur erwartet 5 Argumente. Die Farbe color ist ein Stringwert einer Farbe in den mit PSOUT_PARAM.SYMBOLOLOGY[<n>] definierten Signatur-Bibliotheken. Der Wert von lstyle kann auch ein Pattern aufweisen. Ein Pattern wird definiert mit pattern(l1/l2/l3...) zum Beispiel pattern(5.0/0.5/0.5/0.5). Der 1. Eintrag im Pattern definiert die Länge des 1. Liniensegmentes. Der 2. Eintrag im Pattern definiert die Distanz vom Ende des 1. zum Anfang des 2. Liniensegment. Der 3. Eintrag im Pattern definiert die Länge des 2. Liniensegmentes. Der 4. Eintrag im Pattern definiert die Distanz vom Ende des 2. zum Anfang des 3. Liniensegment. Und so weiter bis zum Ende des Patterns, danach wiederholt sich das Pattern.
- Beispiel** `... => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous,black,0.125.0,2`
- Prozedur** PSOUT_WRITE_POLYGON3 ! a geometrie, s color, i priority
- Beschreibung** Schreibt eine Flächen-Geometrie. Die Prozedur erwartet 3 Argumente. Die Farbe color ist ein Stringwert einer Farbe in den mit PSOUT_PARAM.SYMBOLOLOGY[<n>] definierten Signatur-Bibliotheken.
- Beispiel** `... => PSOUT_WRITE_POLYGON3,IN.GEOM,ForestGreen,1`
- Prozedur** PSOUT_WRITE_TEXT12 ! s text, p geometrie, rot rotation, i halignment, i valignment, s font, r height, s color, r xscale, r slant, r width, i priority
- Beschreibung** Schreibt einen Text. Die Prozedur erwartet 12 Argumente. Die color Farbe ist ein Stringwert einer Farbe in den mit PSOUT_PARAM.SYMBOLOLOGY[<n>] definierten Signatur-Bibliotheken. Der Font font ist ein PostScript-Font oder ein unter ILTOOLS_DIR\system\font*.fd definierter Vektor-Font. Vektor-Fonts werden in Polylines aufgelöst und als Polylines geschrieben. Mit width kann bei solchen Vektor-Fonts die Breite der Polylines als Faktor zur Höhe height des Text definiert werden. Die Breite der Polyline ergibt sich aus height*width.

Beispiel

```
... => PSOUT_WRITE_TEXT12,IN.Objekt.Name,IN.Pos,
      IN.Ori,IN.HAlI,IN.VAlI,standard,2.0,black,,0.0,0.06,4
```

Prozedur

```
PROCEDURE PSOUT_WRITE_SYMBOL7 ! p geometrie, rot rotation, s symbol,
s color, r scale, r width, i priority
```

Beschreibung

Schreibt ein Symbol. Die Prozedur erwartet 7 Argumente. Die Farbe color ist ein Stringwert einer Farbe in den mit PSOUT_PARAM.SYMBOLOLOGY[<n>] definierten Signatur-Bibliotheken. Das Symbol symbol ist ein Stringwert eines Symbols in den mit PSOUT_PARAM.SYMBOLOLOGY[<n>] definierten Signatur-Bibliotheken. Mit width wird die Breite der Polylines des Symbols als Faktor zur Skalierung scale des Symbols definiert. Die Breite der Polylines ergibt sich aus scale*width.

Beispiel

```
... => PSOUT_WRITE_SYMBOL7,IN.Geometrie,0.0,097041,black,1.0,0.1,3
```

Prozedur

```
PROCEDURE PSOUT_WRITE_JPG6 ! a geometrie, r rotation, r width, r
height, s image, i priority
```

Beschreibung

Schreibt ein JPG-Bild. Die Prozedur erwartet 6 Argumente. Die Fläche geometrie für das Bild. Die Rotation rotation des Bildes. Die Breite width und Höhe height des Bildes. Das File image des Bildes. Der Pfad des Bildes kann absolut oder relativ zu ICS_DIR sein.

Beispiel

```
... => PSOUT_WRITE_JPG6,IN.Geometrie,IN.Ori,IN.Width,IN.Height,\plot\iglogo.jpg,-1
```

Prozedur

```
PSOUT_WRITE_CLIP2 ! a geometrie, i priority
```

Beschreibung

Schreibt eine Clip-Flächen-Geometrie. Die Prozedur erwartet 2 Argumente. In der Regel ist nur eine Clip-Fläche zu schreiben. Die Clip-Fläche definiert, welches Gebiet der Daten dargestellt werden soll. Die Clip-Fläche ist nur wirksam for Objekte, die eine höhere Priorität n+i als die Clip-Fläche n haben.

Beispiel

```
... => PSOUT_WRITE_CLIP2,IN.Geometrie,-1001
```

18.10. Skriptbeispiel

```
! Diese ICS Konfiguration schreibt von ilin.mod
! gelesenen Objekte mit psout.mod in eine PostScript
! Datei und wandelt diese in ein PDF-File.
```

```
|LICENSE \license\iltoolspro.lic
```

```
MAP ILIN_PARAM
  INTERLIS_DEF => \models\dm01avch24d.ili
  LOG_TABLE => ON
  TRACE => OFF
  STATISTICS => ON
  CALC_SURFACE => ON
END_MAP
```

```
MAP ILIN_TOPO
  DEFAULT => OFF
END_MAP
```

```
MAP PSOUT_PARAM
```

```

FILE_FORMAT      => pdf
SYMBOLOLOGY1    => \symb\symbology.itf
SYMBOLOLOGY2    => \symb\av.itf
SYMBOLOLOGY3    => \symb\av2.itf
GS_LIB          => c:\ghostscript
STATISTICS      => ON
END_MAP

MAP PLOT_PARAM
TYPE            => PLOT
FORMAT         => A4
FORMATORIENTATION => hoch
ORIGIN         => '675855/245385'
ANGLE          => 45.0
SCALE          => '1:500'
COORDCROSS     => ON
COORDBAND      => ON
SCALEBAND      => ON
ADJUST        => ON
END_MAP

MAP PLOT_LAYOUT
! Format, hoch|quer, Massstab
A0, quer, *    => \plot\a0q500.itf
A0, hoch, *    => \plot\a0h500.itf
A1, quer, *    => \plot\a1q500.itf
A1, hoch, *    => \plot\a1h500.itf
A2, quer, *    => \plot\a2q500.itf
A2, hoch, *    => \plot\a2h500.itf
A3, quer, *    => \plot\a3q500.itf
A3, hoch, *    => \plot\a3h500.itf
A4, quer, *    => \plot\a4q500.itf
A4, hoch, *    => \plot\a4h500.itf
DEFAULT       => \plot\a4h500.itf
END_MAP

MAP PLOT_COORDCROSS_WIDTH
'1:250'       => 50.0
'1:500'       => 50.0
'1:1000'      => 50.0
DEFAULT       => 100.0
END_MAP

MAP PLOT_SCALEBAND_WIDTH
'1:250'       => 12.0
'1:500'       => 20.0
'1:1000'      => 40.0
END_MAP

MAP PLOT_VALUES
PLOT_PARAM.SCALE => OPT.massstab
PLOT_DATE        => OPT.datum
'Demogemeinde'  => OPT.gemeinde
END_MAP

MAP PLOT_WRITE_OBJECT
Plot_Elemente, Border_Flaeche      => OFF
Plot_Elemente, Clipp_Flaeche       => PSOUT_WRITE_CLIP2, IN.Geometrie, -1001

```

```

Plot_Elemente,Flaeche => PSOUT_WRITE_POLYGON3,IN.Geometrie,white,-1005
Plot_Elemente,Linie => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous
Plot_Elemente,Text => PSOUT_WRITE_TEXT12,IN.Text,IN.Geometrie,IN.Or
Plot_Elemente,Bild => PSOUT_WRITE_JPG6,IN.Geometrie,IN.Ori,IN.Width
Plot_Elemente,Symbol_Linie => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous
Plot_Elemente,Symbol_Flaeche => PSOUT_WRITE_POLYGON3,IN.Geometrie,black,-1005

Plot_Generiert,Koordinatenkreuz => PSOUT_WRITE_SYMBOL7,IN.Geometrie,IN.Ori,09705
Plot_Generiert,Koordinatenband_Linie => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous
Plot_Generiert,Koordinatenband_Text => PSOUT_WRITE_TEXT12,IN.Text,IN.Geometrie,IN.Or
Plot_Generiert,Skalierungsband_Clipp_Flaeche => PSOUT_WRITE_POLYGON3,IN.Geometrie,white,10000
Plot_Generiert,Skalierungsband_Border_Linie => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous
Plot_Generiert,Skalierungsband_Block_Flaeche1 => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous
Plot_Generiert,Skalierungsband_Block_Flaeche2 => PSOUT_WRITE_POLYGON3,IN.Geometrie,black,10000
Plot_Generiert,Skalierungsband_Text => PSOUT_WRITE_TEXT12,IN.Text,IN.Geometrie,IN.Or
END_MAP

MAP INPUT_SOURCES
  I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,FixpunkteKategorie3,LFP3Pos => GD_T_1
  I1,FixpunkteKategorie3,LFP3Symbol => GD_S_1
  I1,* => OFF
END_MAP

MAP MACRO
  GD_S_1 => PSOUT_WRITE_SYMBOL7,IN.LFP3Symbol_von.Geometrie,IN.Ori,GP,black,0.6,,50
  GD_T_1 => PSOUT_WRITE_TEXT12,IN.LFP3Pos_von.Nummer,IN.Pos,IN.Ori,IN.HAli,IN.VAli,Helvetica,1.5,b
END_MAP

| INCL \script\plot.mod
| INCL \script\ilTOPO.mod
| INCL \script\psout.mod
| INCL \script\il2ps\il2ps.prg

```

19. Modul SHPOUT - ESRI Shapefile schreiben

19.1. Allgemeines

Mit dem Modul SHPOUT können Objekte in eine ESRI Shapefile Datei geschrieben werden.

Der Modul SHPOUT wird mit:

```
| INCL \script\shpout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

19.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

19.3. Parametermap SHPOUT_PARAM

Folgende Parameter können in der Map SHPOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
FILEPREFIX_GET	o	ON oder OFF, Default: OFF . Definiert ob ein Fileprefix interaktiv abgefragt werden soll.
FILEPREFIX	o	Default: OFF . Definiert ob ein Fileprefix interaktiv abgefragt werden soll.
STROKE_TOL	r	<real> . Stroke-Tolerance zur Auflösung von Kreisbögen in Liniensegmente. Als Stroke-Tolerance können dieselben Werte wie für die ICS-Methode ICS.STROKE verwendet werden (s.a. iG/Script Benutzer- und Referenzhandbuch). Mit OFF werden die Kreisbögen nicht aufgelöst.
DBF_DOS	o	ON oder OFF. Falls ON gesetzt wird, werden DBF-Dateien im DOS-Zeichensatz geschrieben (ArcExplorer). Sonst wird der Windowszeichensatz verwendet (ArcView).
PRJ_FILE	o	<file>, Definiert ein *.prj Template-File mit dem Spatial Referenz System für die Shapefiles. Beispiele sind ILTOOLS_DIR\system\data\SHP_CH1903_LV03.prj und ILTOOLS_DIR\system\data\SHP_CH1903+_LV03.prj. Pfad kann relativ zu ILTOOLS_DIR angegeben werden.
CPG_FILE	o	<Codepage>, Definiert die Codepage (Zeichensatz) für die Sachdaten im dbf-File. Per Default 1252, der Windows-Zeichensatz. Notwendig ab ESRI 10.3/10.4 um z.B. Umlaute richtig darzustellen. Mit dem Wert OFF werden die cpg-Files nicht erzeugt Der Wert des Parameters wird in das cpg-File geschrieben..
DEBUG	r	ON oder OFF, Default = OFF. Debugmodus ein oder aus.
STATISTICS	r	ON oder OFF, Default = OFF. Objektstatistik am Ender der .log Datei anzeigen.

19.4. Objektmodell

Allgemeine Komponenten für jedes OUT-Objekt

Komponente	req/opt	Beschreibung
OUT.FILE(s)	r	Dateiname der Outputdatei <i>ohne</i> Endung .shp.
OUT.TYPE(s)	r	Typ des Outputobjekts (s.a. unten).
OUT.DIMENSION(s)	r	Dimension der Ouputdatei (2D oder 3D).

Zusätzliche Komponenten für OUT.TYPE = 'DBF'

Ein OUT-Objekt vom Typ DBF erzeugt nur eine DBF-Datei.

Zusätzliche Komponenten für OUT.TYPE = 'NULL'

Ein OUT-Objekt vom Typ NULL erzeugt ein NULL-Objekt. Das NULL-Objekt hat keine spezifischen Komponenten.

Zusätzliche Komponenten für OUT.TYPE = 'POINT'

Komponente	req/opt	Beschreibung
OUT.GEOM(p)	r	Punktcoordinate.

Zusätzliche Komponenten für OUT.TYPE = 'MULTIPOINT'

Komponente	req/opt	Beschreibung
OUT.GEOM(1)	r	Liste aus Punktkoordinate(n).

Zusätzliche Komponenten für OUT.TYPE = 'POLYLINE'

Komponente	req/opt	Beschreibung
OUT.STRO- KE_TOL(r)	r	<p>Stroketoleranz. Löst alle Kreisbögen in einer Geometrie in Liniensegmente auf. Ein Kreisbogen wird gleichmässig in Liniensegmente aufgelöst, bis die Toleranz unterschritten ist.</p> <p>Toleranz > 0.0 Die Toleranz ergibt sich aus dem Verhältnis der Pfeilhöhe zum Radius eines Kreisbogens.</p> <p>Toleranz = 0.0 Der Kreisbogen wird in den Anfangs und Endpunkt und den Punkt auf dem Kreisbogen aufgelöst.</p> <p>Toleranz < 0.0 Die Toleranz ist ein absoluter Wert in Meter, der die maximale Pfeilhöhe definiert.</p>
OUT.GEOM(1)	r	Liniengeometrie.

Zusätzliche Komponenten für OUT.TYPE = 'POLYGON'

Komponente	req/opt	Beschreibung
OUT.STRO- KE_TOL(r)	r	Stroketoleranz (s.a. POLYLINE).
OUT.GEOM(a)	r	Flächengeometrie.

19.5. Map für Textsignaturen

Obwohl das SHP Format keine Möglichkeit für die graphische Ausgabe von Text bietet, ist die Darstellung von Text mit SHPOUT trotzdem möglich (s.a. SHPOUT_WRITE_TEXT6). Dazu muss vorgängig die Map TEXT_SYMBOLOGY in der .cfg Datei wie folgt angelegt werden:

```
MAP TEXT_SYMBOLOGY
...
<symbology> => <file>,<font>,<size>,<offset>,<xscale>,<slant>
...
END_MAP
```

Die einzelnen Parameter haben folgende Bedeutung:

<symbology>

Name der definierten Textsymbologie. Der Name der Symbologie muss SHPOUT_WRITE_TEXT6 als Argument übergeben werden.

<file>

Name der Outputdatei.

Name des Fonts. Der Font muss in ICS_DIR\font vorhanden sein.

<size>

Schriftgröße in Benutzereinheiten.

<offset>

Abstand des der Beschriftungsposition vom Einfügepunkt. Der Abstand wird wie eine Punktkoordinate eingegeben, d.h. z.B. 0.0/1.5.

<xscale>

Skalierungsfaktor in x-Richtung.

<slant>

Neigungswinkel in Altgrad.

19.6. Exportierte Prozeduren und Methoden

Prozedur	SHPOUT_OPEN ! [s directory][[]
Beschreibung	Öffnet den SHPOUT Modul auf dem Dateiverzeichnis <directory>. Alle Outputdateien werden in das Verzeichnis <directory> geschrieben. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>'c:\test' SHPOUT_OPEN</code>
Prozedur	SHPOUT_WRITE_DBF1 ! file
Beschreibung	Schreibt nur ein DBF-File. <file> muss <i>ohne</i> Endung und <i>ohne</i> Verzeichnis angegeben werden.
Beispiel	<code>... => SHPOUT_WRITE_DBF1,record</code>
Prozedur	SHPOUT_WRITE_NULL1 ! file
Beschreibung	Schreibt ein NULL Objekt in die Datei <file>. <file> muss <i>ohne</i> Endung und <i>ohne</i> Verzeichnis angegeben werden.
Beispiel	<code>... => SHPOUT_WRITE_NULL1,null_object</code>
Prozedur	SHPOUT_WRITE_POINT3 ! p position,s dimension,s file
Beschreibung	Schreibt ein POINT Objekt in die Datei <file>. Für eine Erklärung der einzelnen Parameter siehe Objektmodell.
Beispiel	<code>... => SHPOUT_WRITE_POINT3,IN.GEOM,2D,point_object</code>
Prozedur	SHPOUT_WRITE_POLYLINE4 ! l geometry, s dimension, r strokto1, s file
Beschreibung	Schreibt ein POLYLINE Objekt in die Datei <file>. Für eine Erklärung der einzelnen Parameter siehe Objektmodell.
Beispiel	<code>... => SHPOUT_WRITE_POLYLINE4,IN.GEOM,2D,0.01,line_object</code>
Prozedur	SHPOUT_WRITE_POLYGON4 ! a geometry, s dimension, r strokto1, s file
Beschreibung	Schreibt ein POLYGON Objekt in die Datei <file>. Für eine Erklärung der einzelnen Parameter siehe Objektmodell.
Beispiel	<code>... => SHPOUT_WRITE_POLYGON4,IN.GEOM,2D,0.01,line_object</code>
Prozedur	SHPOUT_WRITE_TEXT6 ! s txt, p pos, r rot, i hali, i vali, s symbology

Beschreibung	<p>Schreibt ein TEXT Objekt in die Datei <file>. Die einzelnen Paramter haben folgende Bedeutung:</p> <p><txt> Textinhalt.</p> <p><pos> Textposition.</p> <p><rot> Textorientierung.</p> <p><hali> Horizontale Textjustierung in der INTERLIS 1 Kodierung (Wertebereich: 0 .. 2).</p> <p><vali> Vertikale Textjustierung in der INTERLIS 1 Kodierung (Wertebereich: 0 .. 4).</p> <p><symbology> Für eine Erklärung des Symbologieparameters siehe Objektmodell.</p>
Beispiel	<pre>... => SHPOUT_WRITE_TEXT6, IN.TXT, IN.POS, IN.ORI, IN.HALI, IN.VALI, text_object</pre>
Prozedur	SHPOUT_CLOSE ! [][]
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<pre>SHPOUT_CLOSE</pre>
Methode	SHPOUT.CREATE_FILE [s Datei, s Typ, s Dimension, m Attribute][]
Beschreibung	Erzeugt eine neue Shapedatei. Die einzelnen Parameter haben folgende Bedeutung: <Datei>: Name der Shapedatei ohne Endung. <Typ>: Objekttyp der Shapedatei (NULL, POINT, POLYLINE, POLYGON). <Dimension>: 2D oder 3D (ohne Bedeutung für NULL Objekte). <Attribute>: Map in welcher die Objektattribute beschrieben sind.
Beispiel	<pre>'Punkte' 'POINT' '2D' &POINT_REC SHPOUT.CREATE_FILE</pre>

19.7. Skriptbeispiel

```
! Diese ICS Konfiguration kopiert alle von shpin.mod
! gelesenen Objekte mit shpout.mod in in eine .shp Datei.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG => FILE ! FILE | FILES | DIRECTORY | STRING | ODBC
  MESSAGE => 'Enter .shp Input File'
  FILE_FILTER => shp
  FILE_EXISTS => TRUE
  OPT => input
END_MAP
```

```
MAP USER_INPUT2
```

```

DIALOG => DIRECTORY ! FILE | FILES | DIRECTORY | STRING | ODBC
MESSAGE => 'Enter .shp Output Directory'
FILE_FILTER => *
OPT => output
END_MAP

MAP SHPIN_PARAM
  STATISTICS => ON
  DEBUG      => OFF
END_MAP

MAP SHPOUT_PARAM
  STATISTICS => ON
  PRJ_FILE  => \data\SHPIN.CH1903_LV03.prj
  DEBUG     => OFF
END_MAP

MAP INPUT_SOURCES
  I1 => SHPIN,OPT.input
END_MAP

MAP INOUT
  I1 => COPY_INOUT0
END_MAP

|INCL \script\shpin.mod
|INCL \script\shpout.mod

PROCEDURE COPY_INOUT0
  ! copy IN map
  &IN &OUT MAPCOPY
  ! copy all attributes
  &SHPIN_REC &SHPOUT_REC MAPCOPY
  SHPOUT_WRITE_OBJECT
END_PROCEDURE

|INCL \script\run1.prg

```

20. Modul **SQLITEOUT** - SQLite-Datenbank schreiben

20.1. Allgemeines

Mit dem Skriptmodul können Objekte in eine SQLite-Datenbank geschrieben werden.

Insbesondere werden die Spezifikationen von **OGC Geopackages** (Vektor-Geometrien) unterstützt.

Der Modul wird mit:

```
|INCL \script\sqliteout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

20.2. Parametermap SQLITEOUT_PARAM

Folgende Parameter können in der Map SQLITEOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltet werden. Sie dazu mehr unter dem Kapitel Datasets.
SRID	o	<STRING>. Die Nummer des EPSG Spatial Reference System. Die SRID muss in der Table gpkg_spatial_ref_sys enthalten sein. 2056 für LV95, 21781 für LV03.
STROKE	o	<real> . Stroke-Tolerance zur Auflösung von Kreisbögen in Liniensegmente. Als Stroke-Tolerance können dieselben Werte wie für die ICS-Methode ICS.STROKE verwendet werden (s.a. iG/Script Benutzer- und Referenzhandbuch). Mit OFF werden die Kreisbögen nicht aufgelöst.
ST_METADATA	o	ON oder OFF, Default = OFF. Definiert, ob die ST_* Metadaten in der Datenbank erzeugt werden sollen.

20.3. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tablename des OUT Objekts.
OUT.<Attribut>(s)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Typ des Werts muss mit dem Typ des Attributs in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden.

Dieses Objektmodell gilt für die Anwendung mit der Prozedur SQLITEOUT_WRITE_OBJECT0. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfiguration mit Prozeduren und Methoden vorgängig erzeugt werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, so beachten Sie das Kapitel mit den Record-Definitionen und die Prozedur SQLITEOUT_WRITE_RECORD1.

20.4. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank angelegt werden. Die Record Definitionen werden von der Prozedur DBOUT_WRITE_RECORD1 verarbeitet. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<n>
  TABLE => <Table-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_ <n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix RECORD_ beginnen und eine eindeutige Nummer <n> für die Record Definition beinhalten.

TABLE

Diese Komponente ist required und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente TABLE die Datenbank-Tabelle.

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

Alle Datenbanken**CHAR(<length>)**

Stringtypen sind immer als CHAR mit der Länge <length> zu definieren..

NUMBER(p,s)

Number-Type.

INTEGER

Integer-Type.

GEOMETRY(<type>;<dimension>;<HASM>)

Geometrien müssen als Type GEOMETRY definiert werden. Dabei sind folgende weiteren Definitionen notwendig.

<type>

Type der Geometrie, einer der Werte: point|line|area.

<dimension>

Dimension der Geometrie, einer der Werte: 2D|3D.

<HASM>

Geometrie besitzt die Measure-Dimension, einer der Werte: TRUE|FALSE.



SQLite OGC Geopackage erlaubt nur eine Geometrie-Definition pro Tabelle. Deshalb kann pro Record-Definition nur ein Geometrie-Attribut definiert werden.



Um mögliche Einschränkungen von SDE zu umgehen, ist es empfehlenswert, den Geometrie-Attributen den Name SHAPE zu vergeben.



HASM wird zur Zeit nicht unterstützt.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können absolute Werte oder ICS-Variablen, die einen Wert beinhalten - z.B. IN.OBJID - verwendet werden.

Beispiel einer Record Definition.

```
MAP RECORD_1
  TABLE => Bodenbedeckung_BoFlaeche_Area
  OBJID => CHAR(10), IN.OBJID
  Entstehung => CHAR(10), IN.Entstehung.OBJID
```

```

Qualitaet => CHAR(30),IN.Qualitaet
Art => INTEGER,IN.Art
Art_TXT => CHAR(47),IN.Art_TXT
Herkunft => CHAR(30),IN.Herkunft
GEOM => GEOMETRY(area;2D),IN.GEOM
END_MAP

```

20.5. Datasets

Mit dem Parameter `DBOUT_PARAM.DATASET => ON` kann der Modul veranlasst werden, die Daten in Datasets zu verwalten. In der Regel werden verschiedene Datasets in einer Datenbank geschrieben. Die einzelnen Datasets können danach als Subset des Daten nachgeführt oder gelöscht werden. Typischerweise bildet ein INTERLIS-File ein solches Dataset. Falls der Dataset-Parameter eingeschaltet ist, wird in der Datenbank folgendes angelegt:

```

CREATE TABLE GS_FILE (
  FileID      INTEGER      -- Schlüssel für Dataset
  ModelName   CHAR(255)    -- Modellname (INTERLIS)
  FileName    CHAR(255)    -- Filenamen
  DateFile    INTEGER      -- YYYYMMDD Datum des Files
  DateUpload  INTEGER      -- YYYYMMDD Datum des lesen in die Datenbank
  UserUpload  CHAR(255)    -- eventuell ein Username
);

CREATE TABLE <Data-Table> (
  GS_FileID   INTEGER      -- Fremdschlüssel für Dataset
  :
);

```

Die Tabelle `GS_FILE` verwaltet die Datasets. Jedes Dataset erhält einen eindeutigen Schlüssel im Attribut `FileID`. Jede angelegte Tabelle für die Daten erhält ein Attribut `GS_FileID`, das den Fremdschlüssel des Datasets beinhalten.

Wird ein Dataset (File) das erste Mal in die Datenbank geschrieben (`INSERT`), wird der Eintrag in `GS_FILE` generiert, und jeder Daten-Record erhält den Fremdschlüssel des Datasets.

Wird ein Dataset (File) ein nächstes Mal in die Datenbank geschrieben (`UPDATE`), wird der Eintrag in `GS_FILE` mit den Daten nachgeführt, die bestehenden Daten des Datasets in der Datenbank gelöscht und die neuen Daten in die Datenbank geschrieben.

20.6. Datenbank Modellgenerierung mit `CONFIG_PARAM.GENERATE_MODEL`

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell erstellt werden soll, so ist der Parameter `CONFIG_PARAM.GENERATE_MODEL` auf `ON` zu setzen:

```

MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP

```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell angelegt, falls es nicht schon angelegt wurde.

20.7. Exportierte Prozeduren und Methoden

Prozedur	<code>SQLITEOUT_OPEN [s input][]</code>
Beschreibung	Öffnet eine Datenbank in Abhängigkeit von <input>. Existiert die Datenbank nicht, so wird diese kreiert. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>OPT.input SQLITEOUT_OPEN</code>
Prozedur	<code>SQLITEOUT_WRITE_OBJECT0</code>
Beschreibung	Schreibt ein Objekt in die Datenbank. Das OUT-Objekt muss gemäss dem Objektmodell gefüllt sein.
Beispiel	<code>SQLITEOUT_WRITE_OBJECT0</code>
Prozedur	<code>SQLITEOUT_WRITE_RECORD1 ! s recordname</code>
Beschreibung	Schreibt ein Objekt definiert in <recordname> in die Datenbank. <recordname> ist der Name einer Record-Definition (siehe weiter oben). Je nach den gesetzten Parametern führt die Prozedur beim ersten Aufruf für eine Record-Definition folgende Aktionen auf der Datenbank aus: <ol style="list-style-type: none"> 1. Die Tabelle wird erzeugt, falls definiert durch Parameter. 2. Das Objekt wird in die Datenbank geschrieben.
Beispiel	<code>... => SQLITEOUT_WRITE_RECORD1,RECORD_1</code>
Prozedur	<code>SQLITEOUT_CLOSE [] []</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>SQLITEOUT_CLOSE</code>

20.8. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,
! erzeugt eine Tabelle für LFP's in der Datenbank
! und schreibt die LFP's in die Tabelle.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter INTERLIS Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT         => input
END_MAP
```

```
MAP USER_INPUT2
  DIALOG      => FILE
  MESSAGE     => 'Enter SQLITE Output File'
  FILE_FILTER => gpkg;db
```

```
FILE_EXISTS => FALSE
OPT         => output
END_MAP

MAP ILIN_PARAM
INTERLIS_DEF => '\models\DM01AVCH24LV95D.ili'
STATISTICS  => ON
CALC_SURFACE => ON
ENUM_TO_TEXT => ON
TRACE       => OFF
END_MAP

MAP ILIN_TOPO
DEFAULT => OFF
END_MAP

MAP SQLITEOUT_PARAM
STATISTICS  => ON
DATASET     => ON
SRID        => 2056
END_MAP

MAP RECORD_1
TABLE => FixpunkteKategoriel_LFP1
OBJID => CHAR(32),IN.OBJID
Entstehung => CHAR(32),IN.Entstehung.OBJID
NBIdent => CHAR(12),IN.NBIdent
Nummer => CHAR(12),IN.Nummer
Geometrie => GEOMETRY(point;2D;FALSE),IN.Geometrie
HoeheGeom => NUMBER(7,3),IN.HoeheGeom
LageGen => NUMBER(4,1),IN.LageGen
LageZuv => INTEGER,IN.LageZuv
LageZuv_TXT => CHAR(4),IN.LageZuv_TXT
HoeheGen => NUMBER(4,1),IN.HoeheGen
HoeheZuv => INTEGER,IN.HoeheZuv
HoeheZuv_TXT => CHAR(4),IN.HoeheZuv_TXT
Begehbarkeit => INTEGER,IN.Begehbarkeit
Begehbarkeit_TXT => CHAR(14),IN.Begehbarkeit_TXT
Punktzeichen => INTEGER,IN.Punktzeichen
Punktzeichen_TXT => CHAR(17),IN.Punktzeichen_TXT
END_MAP

MAP INPUT_SOURCES
I1 => ILTOPO,OPT.input
END_MAP

MAP INOUT
I1 => IN.TOPIC,IN.TABLE
I1,FixpunkteKategoriel,LFP1 => R_1,RECORD_1
I1,* => OFF
END_MAP

| INCL \script\iltopo.mod
| INCL \script\sqliteout.mod
| INCL \script\run1.prg
```

21. Modul TXTOUT - Textdateien schreiben

21.1. Allgemeines

Mit dem Modul TXTOUT können Objekte in eine Textdatei geschrieben werden.

Der Modul TXTOUT wird mit:

```
| INCL \script\txtout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

21.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

21.3. Parametermap TXTOUT_PARAM

Folgende Parameter können in der Map TXTOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
TRACE	o	ON oder OFF, Default = OFF. Tracemodus ein oder aus.
STATISTICS	o	ON oder OFF, Default = OFF. Objektstatistik am Ender der .log Datei anzeigen.
HEADER	o	ON oder OFF, Default = ON. Soll Zeilenheader geschrieben werden.
DELIMITER	o	string, Default = 9. ASCII Code oder Character für Trennzeichen. 9 = Tabulator.
TEXTENCLOSURE	o	string, ASCII Code oder Character für Einfassung
OUTPUT_APPEND	o	ON oder OFF, Default = OFF. Soll der Output an ein bestehendes Outputfile angehängt werden.

21.4. Objektmodell

Der Modul verlangt für jedes OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.FILE(s)	r	File für Objekt.
OUT.<Attribute>(*)	o	Attribut mit Wert.

Dieses Objektmodell gilt für die Anwendung mit der Prozedur TXTOUT_WRITE_OBJECT und TXTOUT_WRITE_OBJECT1.

21.5. Record Definitionen

Mit Record Definitionen können nicht Objekte vereinfacht geschrieben werden. Die Record Definitionen werden von der Prozedur TXTOUT_WRITE_RECORD1 verarbeitet. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<name>
  FILE => <FILE-Name>
  :
  <Attribute-Name> => <Attribute-Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<name>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix RECORD_ beginnen und einen eindeutigen Namen <name> für die Record Definition beinhalten.

FILE

Diese Komponente ist required und definiert das Output-File.

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen des Files. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können absolute Werte oder ICS-Variablen, die einen Wert beinhalten - z.B. IN.OBJID - verwendet werden.

Beispiel einer Record Definition.

```
MAP RECORD_LFP1
  FILE      => LFP1.txt
  Art       => IN.TABLE
  Nummer    => IN.Nummer
  X         => IN.X
  Y         => IN.Y
  Z         => IN.Z
END_MAP
```

21.6. Exportierte Prozeduren und Methoden

Prozedur **TXTOUT_OPEN ! [s file][]**

Beschreibung Erzeugt die Textdatei <file>. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel `'c:\test.txt' TXTOUT_OPEN`

Prozedur **TXTOUT_WRITE_OBJECT ! [][]**

Beschreibung Schreibt das OUT-Objekt in die Outputdatei.

Beispiel `&OUT MAPCLEAR
'data.txt' => OUT.FILE
'hello world => OUT.Name
TXTOUT_WRITE_OBJECT`

Prozedur **TXTOUT_WRITE_OBJECT1 ! file**

Beschreibung Schreibt das OUT-Objekt in die Outputdatei.

Beispiel `... => TXTOUT_WRITE_OBJECT1,data.txt`

Prozedur **TXTOUT_WRITE_RECORD1 ! record**

Beschreibung Schreibt einen Record.

Beispiel ... => TXTOUT_WRITE_OBJECT1,RECORD_1

Prozedur TXTOUT_CLOSE ! [][]

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel TXTOUT_CLOSE

21.7. Skriptbeispiel

```
! Dieser Skript uebersetzt Fixpunkte einer INTERLIS
! Datei in eine Textdatei. Die erste Zeile der Textdatei
! enthaelt alle Attributnamen. Die weiteren Zeilen enthalten
! die Punktfelder getrennt durch Tabulatorzeichen (ASCII = 9).
```

```
MAP USER_INPUT1
  DIALOG => FILE
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP USER_INPUT2
  DIALOG => FILE
  MESSAGE => 'Enter .txt Output File'
  FILE_FILTER => txt
  FILE_EXISTS => FALSE
  OPT => output
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\DM01AVCH24D.ili
  STATISTICS => ON
  ENUM_TO_TEXT => ON
END_MAP

MAP TXTOUT_PARAM
  STATISTICS => ON
  DELIMITER => 9          ! ASCII-Code or Character
  HEADER => ON           ! ON|OFF
END_MAP

MAP RECORD_LFP1
  FILE => LFP1.txt
  Art => IN.TABLE
  Nummer => IN.Nummer
  X => IN.X
  Y => IN.Y
  Z => IN.Z
END_MAP

MAP RECORD_LFP2
  FILE => LFP2.txt
```

```
Art      => IN.TABLE
Nummer   => IN.Nummer
X        => IN.X
Y        => IN.Y
Z        => IN.Z
END_MAP

MAP RECORD_LFP3
FILE     => LFP3.txt
Art      => IN.TABLE
Nummer   => IN.Nummer
X        => IN.X
Y        => IN.Y
Z        => IN.Z
END_MAP

MAP INPUT_SOURCES
I1 => ILIN,OPT.input
END_MAP

MAP INOUT
I1 => IN.TOPIC,IN.TABLE
I1,FixpunkteKategorie1,LFP1 => LFP_prepare_0,R_1,RECORD_LFP1
I1,FixpunkteKategorie2,LFP2 => LFP_prepare_0,R_1,RECORD_LFP2
I1,FixpunkteKategorie3,LFP3 => LFP_prepare_0,R_1,RECORD_LFP3
I1,* => OFF
END_MAP

MAP MACRO ! macros
R_1 => TXTOUT_WRITE_RECORD1
DIN => DISPLAY_OBJECT1,IN
DOUT => DISPLAY_OBJECT1,OUT
END_MAP

PROCEDURE LFP_prepare_0
IN.Geometrie POINTX '3' ROUND => IN.X
IN.Geometrie POINTY '3' ROUND => IN.Y
IN.HoeheGeom      '3' ROUND => IN.Z
END_PROCEDURE

| INCL \script\ilin.mod
| INCL \script\txtout.mod
| INCL \script\run1.prg
```

22. Modul XLSOUT - MS Excel schreiben

22.1. Allgemeines

Mit dem Skriptmodul können Objekte in eine Excel Tabelle geschrieben werden.

Der Modul wird mit:

```
| INCL \script\xlsout.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

22.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

22.3. Parametermap DB_PARAM

Folgende Parameter können in der Map DB_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
TRACE	o	ON oder OFF, Default = OFF. Für jedes gelesene Objekt eine Zeile ausgeben.

22.4. Parametermap DBOUT_PARAM

Folgende Parameter können in der Map DBOUT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
CREATE_TABLE	o	ON, OFF, Default = OFF. Definiert ob die Tables definiert mit Record-Definitionen (siehe weiter unten) in der Datenbank erzeugt werden sollen. Mit OFF werden die Tabellen nicht erzeugt. Mit ON werden die Tabellen erzeugt und zwar zum Zeitpunkt, wenn das erste Objekt in die Tabelle geschrieben wird. Tables in die keine Objekte geschrieben werden, werden auch nicht erzeugt.
DELETE_OLD	o	ON, OFF oder DROP, Default = OFF. Löscht bestehende Daten in Tables definiert mit Record-Definitionen (siehe weiter unten). Mit OFF werden keine Daten gelöscht. Mit ON werden die Daten über ein delete-sql-statement gelöscht. Mit DROP wird die Table und damit die Daten gelöscht. Zusammen mit DROP und dem Parameter CREATE_TABLE => ON werden die Tables gelöscht und wieder erzeugt.
BATCH	o	ON oder OFF, Default = OFF. Mit OFF werden die sql-statements direkt auf der Datenbank ausgeführt. Mit OFF werden die sql-statements in ein Batchfile geschrieben. Mit dem Parameter BATCH_FILE wird das Batchfile definiert.
BATCH_OUTPUT_DIR	o	<directory>. Definiert ein Output-Directory für Batchfiles. Mit diesem Parameter kann das Output-Directory für Batchfiles definiert werden, falls der Parameter BATCH = ON definiert ist. Batchfiles können sein ein File mit SQL-Statements oder in Kombination mit dem Oracle Output Modul die SQLLOADER-Bulkfiles. Ist dieser Parameter nicht gesetzt, so wird das Output-Directory aus einem eventuellen Input-File definiert in OPT.input bestimmt. Ist kein Input-File definiert, so ist das Output-Directory iltools\data\ics.sql.
BATCH_FILE	o	<file>. Definiert das Batchfile. Mit diesem Parameter kann das Batchfile definiert werden, falls der Parameter BATCH = ON definiert ist. Ist dieser Parameter nicht gesetzt, so wird das Batchfile aus einem eventuellen Input-File definiert in OPT.input mit der Endung .sql bestimmt. Ist kein Input-File definiert, so ist das Batchfile iltools\data\ics.sql als definiert. Das Batchfile beinhaltet SQL-Statements, um die transferierten Daten mittels SQL in eine Datenbank zu importieren.

SQLTRACE	o	ON oder OFF, Default = OFF. Zeigt als sql-statements im Logfile an.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
DATASET	o	ON oder OFF, Default = OFF. Definiert, ob die in die Datenbank geschriebenen Daten als Datasets verwaltet werden. Sie dazu mehr unter dem Kapitel Datasets. Der Wert ON hebt den Parameter DELETE_OLD auf.

22.5. Parametermap XLSOUT_PARAM

In dieser Parameter Map sind zur Zeit keine Parameter definierbar.

Parameter	req/opt	Beschreibung
-----------	---------	--------------

22.6. Objektmodell

Der Modul verlangt pro OUT-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
OUT.TABLE(s)	r	Tablename des OUT Objekts.
OUT.<Attribut>(s)	o	Attributname der Tabelle. Der Wert der Komponente beinhaltet den Wert für die Datenbank. Der Typ des Werts muss mit dem Typ des Attributs in der Datenbank übereinstimmen. Es können beliebig viele Attribute definiert werden.

Dieses Objektmodell gilt für die Anwendung mit der Prozedur XLSOUT_WRITE_OBJECT0. Die Tabellen mit den Attributen müssen in der Datenbank bereits bestehen oder in einer Konfiguration mit Prozeduren und Methoden vorgängig erzeugt werden. Falls die Datenbankstrukturen automatisiert mit dem Modul erstellt werden sollen, so beachten Sie das Kapitel mit den Record-Definitionen und die Prozedur XLSOUT_WRITE_RECORD1.

22.7. EXCEL Tabelle vorbereiten

Tabelle gesamthaft schreiben

Wenn der Modul eine Excel-Tabelle vollständig schreiben soll, muss die Excel-Tabelle nach EXCEL-ODBC-Konvention wie folgt angesprochen werden:

```
[<table>$]
```

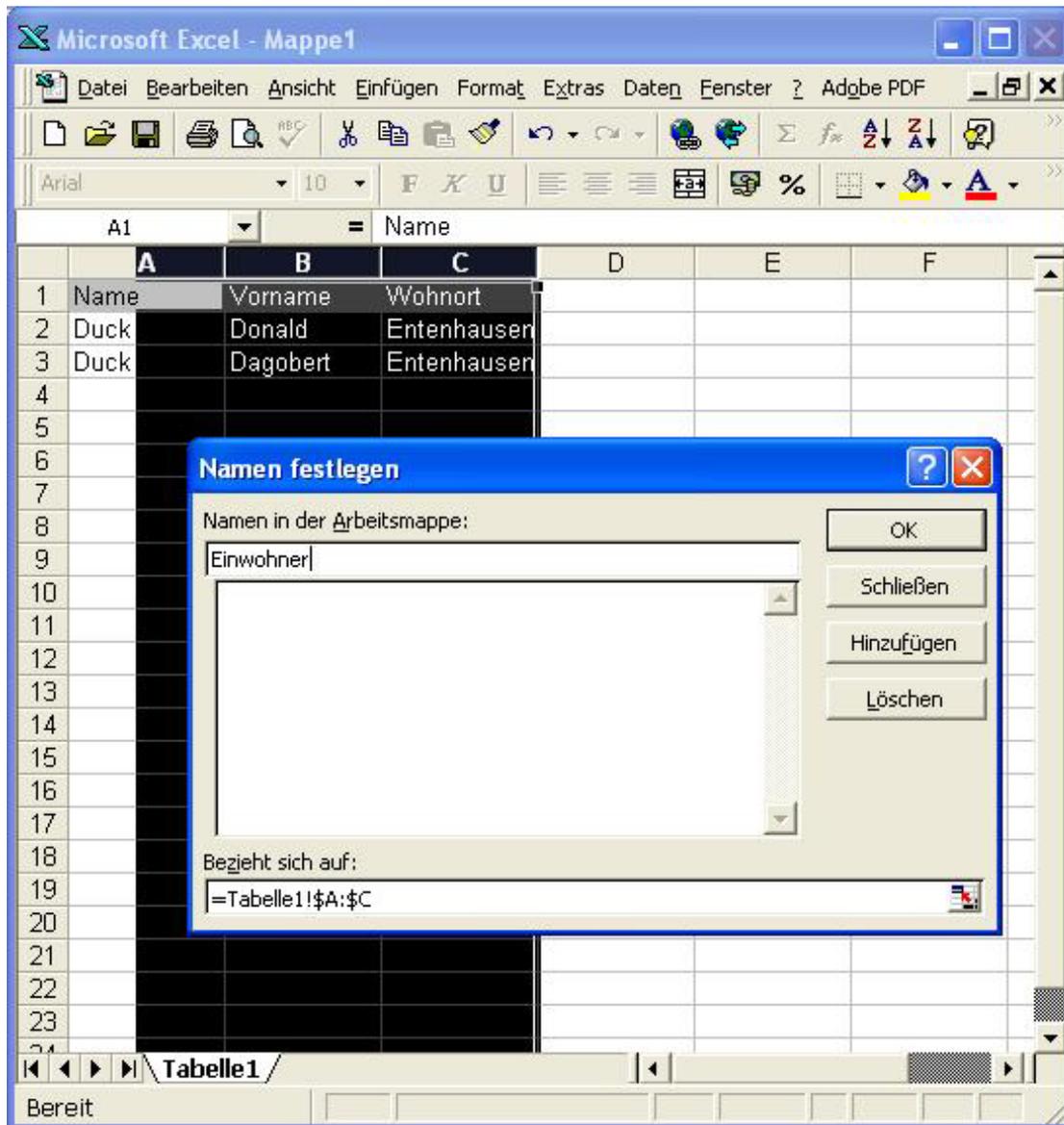
Beispiel:

```
MAP RECORD_1
  TABLE => [Tabelle1$]
  :
END_MAP
```

Tabelle tabellarisch schreiben

Damit der Modul Daten tabellarisch - Teile einer Tabelle - in eine Excel-Tabelle schreiben kann, ist die Excel Tabelle wie folgt unter Excel vorzubereiten.

Abbildung B.1.



Spaltennamen

In der ersten Zeile sind die Spaltennamen einzutragen.

Tabellennamen

Die Spalten sind zu markieren.

Über Einfügen > Namen > Festlegen ist dem markierten Bereich ein Namen zu vergeben.

Der markierte Bereiche wird über diesen Namen vom Modul XLSOUT angesprochen.

z.B.

```
MAP RECORD_1
  TABLE => Einwohner
  :
END_MAP
```

22.8. Record Definitionen

Mit Record Definitionen können nicht nur Objekte in die Datenbank geschrieben werden, sondern auch Datenbank-Tabellen automatisiert in der Datenbank angelegt werden. Die Record Definitionen werden von der Prozedur XLSOUT_WRITE_RECORD1 verarbeitet. Eine Record Definition sieht wie folgt aus.

```
MAP RECORD_<n>
  TABLE => <Table-Name>
  :
  <Attribute-Name> => <Attribute-Type>,<Attribute-Value>
  :
END_MAP
```

Die einzelnen Bestandteile einer Record-Definition sind:

RECORD_<n>

Eine Record Definition ist eine Map. Der Map-Name muss mit dem Prefix RECORD_ beginnen und eine eindeutige Nummer <n> für die Record Definition beinhalten.

TABLE

Diese Komponente ist required und definiert die Datenbank-Tabelle.

<Table-Name>

Definiert als Wert der Komponente TABLE die Datenbank-Tabelle.

<Attribute-Name>

Definiert als Komponente einen Attribut-Namen der Tabelle. Es können beliebig viele Attribute als Komponenten definiert werden.

<Attribute-Type>

Definiert den Attribut-Type in der Datenbank. Im Normalfall können der Datenbank bekannt Typen verwendet werden. Spezialtypen einer Datenbank werden eventuell noch nicht unterstützt. Solche Spezialtypen werden bei Bedarf und auf Anfrage implementiert. Bei den Typen ist auf folgendes zu achten.

Alle Datenbanken

CHAR(<length>)

Stringtypen sind immer als CHAR mit der Länge <length> zu definieren. Je nach Datenbank wird der Typ vom Modul in einen der Datenbank bekannten Type umgewandelt z.B. Excel als TEXT.

Die length darf maximal 255 betragen.

NUMBER

Number-Typen für reelle und ganze Zahlen.

<Attribut-Value>

Definiert den Wert für das Attribut. Als Wert können absolute Werte oder ICS-Variablen, die einen Wert beinhalten - z.B. IN.OBJID - verwendet werden.

Beispiel einer Record Definition.

```
MAP RECORD_1
  TABLE => Bodenbedeckung_BoFlaeche_Area
  OBJID => CHAR(10),IN.OBJID
  Entstehung => CHAR(10),IN.Entstehung.OBJID
  Geometrie => DB_GEOMETRY(point;2D),IN.Geometrie
  Qualitaet => CHAR(30),IN.Qualitaet
  Art => INTEGER,IN.Art
```

```

Art_TXT => CHAR(47),IN.Art_TXT
Herkunft => CHAR(30),IN.Herkunft
END_MAP

```

22.9. Datasets

Mit dem Parameter `DBOUT_PARAM.DATASET => ON` kann der Modul veranlasst werden, die Daten in Datasets zu verwalten. In der Regel werden verschiedene Datasets in einer Datenbank geschrieben. Die einzelnen Datasets können danach als Subset des Daten nachgeführt oder gelöscht werden. Typischerweise bildet ein INTERLIS-File ein solches Dataset. Falls der Dataset-Parameter eingeschaltet ist, wird in der Datenbank folgendes angelegt:

```

CREATE TABLE GS_FILE (
  FileID      INTEGER      -- Schlüssel für Dataset
  ModelName   CHAR(255)    -- Modellname (INTERLIS)
  FileName    CHAR(255)    -- Filenamen
  DateFile    INTEGER      -- YYYYMMDD Datum des Files
  DateUpload  INTEGER      -- YYYYMMDD Datum des lesen in die Datenbank
  UserUpload  CHAR(255)    -- eventuell ein Username
);

CREATE TABLE <Data-Table> (
  GS_FileID   INTEGER      -- Fremdschlüssel für Dataset
  :
);

```

Die Tabelle `GS_FILE` verwaltet die Datasets. Jedes Dataset erhält einen eindeutigen Schlüssel im Attribut `FileID`. Jede angelegte Tabelle für die Daten erhält ein Attribut `GS_FileID`, das den Fremdschlüssel des Datasets beinhaltet.

Wird ein Dataset (File) das erste Mal in die Datenbank geschrieben (`INSERT`), wird der Eintrag in `GS_FILE` generiert, und jeder Daten-Record erhält den Fremdschlüssel des Datasets.

Wird ein Dataset (File) ein nächstes Mal in die Datenbank geschrieben (`UPDATE`), wird der Eintrag in `GS_FILE` mit den Daten nachgeführt, die bestehenden Daten des Datasets in der Datenbank gelöscht und die neuen Daten in die Datenbank geschrieben.

Soll ein Dataset (File) aus der Datenbank gelöscht werden (`DELETE`), so steht folgende Konfiguration zur Verfügung:

```
ILTOOLS_DIR\system\script\il2db\dbdatasetdelete.cfg
```

Diese Konfiguration löscht die Daten eines Datasets und den Eintrag des Datasets in `GS_FILE`.

22.10. Datenbank Modellgenerierung mit `CONFIG_PARAM.GENERATE_MODEL`

Falls basierend auf den Record-Definitionen vor einem Datentransfer das gesamte Datenbankmodell erstellt werden soll, so ist der Parameter `CONFIG_PARAM.GENERATE_MODEL` auf `ON` zu setzen und das Script `il2db.lib` zu includen:

```

MAP CONFIG_PARAM
  GENERATE_MODEL => ON
END_MAP
:
|INCL \script\il2db\il2db.lib

```

Entsprechend den Record-Definitionen wird vor einem Datentransfer das Datenbankmodell angelegt, falls es nicht schon angelegt wurde.

22.11. Exportierte Prozeduren und Methoden

Prozedur	<code>XLSOUT_OPEN [s file][[]]</code>
Beschreibung	Öffnet eine Excel File. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>OPT.output XLSOUT_OPEN</code>
Prozedur	<code>XLSOUT_WRITE_OBJECT0</code>
Beschreibung	Schreibt ein Objekt in eine Excel Tabelle. Das OUT-Objekt muss gemäss dem Objektmodell gefüllt sein.
Beispiel	<code>XLSOUT_WRITE_OBJECT0</code>
Prozedur	<code>XLSOUT_WRITE_RECORD1 ! s recordname</code>
Beschreibung	Schreibt ein Objekt definiert in <recordname> in die Datenbank. <recordname> ist der Name einer Record-Definition (siehe weiter oben). Je nach den gesetzten Parametern führt die Prozedur beim ersten Aufruf für eine Record-Definition folgende Aktionen auf der Datenbank aus: <ol style="list-style-type: none"> 1. Die Tabelle oder deren Inhalt wird gelöscht, falls definiert durch Parameter. 2. Die Tabelle wird erzeugt, falls definiert durch Parameter. 3. Das Objekt wird in die Datenbank geschrieben.
Beispiel	<code>... => XLSOUT_WRITE_RECORD1,RECORD_1</code>
Prozedur	<code>XLSOUT_CLOSE [[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>DBOUT_CLOSE</code>

Neben diesen Prozeduren des Modules stehen auch die Methoden der Klasse DB zur Verfügung (s.a. iG/Script Benutzer- und Referenzhandbuch).

22.12. Skriptbeispiel

```
! Diese ICS Konfiguration liest ein INTERLIS File,
! erzeugt eine Tabelle für LFP's in Excel
! und schreibt die LFP's in die Tabelle.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG      => FILE
  MESSAGE     => 'Enter INTERLIS Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
```

```

    OPT          => input
END_MAP

MAP USER_INPUT2
    DIALOG       => FILE
    MESSAGE      => 'Enter Excel Output File'
    FILE_FILTER  => xls
    FILE_EXISTS  => FALSE
    OPT          => output
END_MAP

MAP ILIN_PARAM
    INTERLIS_DEF => \models\Grunddatensatz.ili
    STATISTICS   => ON
    CALC_SURFACE => ON
    ENUM_TO_TEXT => ON
    TRACE        => OFF
END_MAP

MAP ILIN_TOPO
    DEFAULT      => OFF
END_MAP

MAP DB_PARAM
    TRACE        => OFF
END_MAP

MAP DBOUT_PARAM
    STATISTICS   => ON
    CREATE_TABLE => ON
    DATASET      => ON
END_MAP

MAP XLSOUT_PARAM
END_MAP

MAP RECORD_1
    TABLE       => Fixpunkte_LFP
    OBJID        => CHAR(10),IN.OBJID
    ENTSTEHUNG  => CHAR(10),IN.Entstehung.OBJID
    NUMMER       => CHAR(12),IN.Nummer
    GEOMETRIE   => DB_GEOMETRY(point;3D),IN.Geometrie
    LAGEZUV_TXT => CHAR(4),IN.LageZuv_TXT
    HOEHEZUV_TXT => CHAR(4),IN.HoeheZuv_TXT
    BEGEHBARKEIT_TXT => CHAR(14),IN.Begehbarkeit_TXT
    SYMBOLORI   => NUMBER,IN.SymbolOri
    ART_TXT     => CHAR(4),IN.Art_TXT
    HERKUNFT    => CHAR(30),IN.Herkunft
END_MAP

MAP INPUT_SOURCES
    I1           => ILTOPO,OPT.input
END_MAP

MAP INOUT
    I1           => IN.TOPIC,IN.TABLE
    I1,Fixpunkte,LFP => XLSOUT_WRITE_RECORD1,RECORD_1
    I1,*         => OFF

```

```
END_MAP
|INCL \script\iltopo.mod
|INCL \script\xlsout.mod
|INCL \script\run1.prg
```

C. Verarbeitungs Module

1. Einleitung

In diesem Anhang sind alle Verarbeitungs Module und ihre Prozeduren bzw. Methoden beschrieben welche zusammen mit dem RUN1 Algorithmus benutzt werden können. Verarbeitungsmodul sind Module welche sich nicht eindeutig in die Kategorie Inputmodul oder Outputmodul einteilen lassen (z.B. weil sie zugleich Input- wie auch Outputmodul sind).

2. Modul GEOINDEX - Geometrie Index

2.1. Allgemeines

Mit dem Modul GEOINDEX werden Objekte mit Geometrien in einen Geometrie Index geschrieben. Über geometrische Abfragen können diese Objekte schnell aufgefunden werden.

GEOINDEX wird mit:

```
|INCL \script\geoindex.mod
```

verfügbar gemacht.

2.2. Parametermap GEOINDEX_PARAM

Folgende Parameter können in der Map GEOINDEX_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
PAGESIZE	o	<real>. Geometrien werden in einem Raster (Pages) indexiert. Mit diesem Parameter wird die Rastergröße definiert. Je nach der Art der Geometrien und der geplanten Verarbeitung ist eine entsprechende Rastergröße optimal.
DISKMEMORY	o	ON oder OFF, Default = OFF. Per Default speichert das Modul die indexierten Objekte und Geometrie im Arbeitsspeicher. Bei großen Datenmengen kann eventuell der Arbeitsspeicher nicht ausreichen. Mit diesem Parameter kann gesetzt werden, dass die indexierten Objekte und Geometrie auf der Disk gespeichert werden. Dies Verarbeitung entlastet den Arbeitsspeicher ist aber etwas langsamer.

2.3. Objektmodell

Mit dem Modul werden Objekte mit Geometrien in einen Index geschrieben. Die Objekte können beliebig sein und werden analog wieder gelesen.

2.4. Exportierte Prozeduren und Methoden

Bevor Abfragen auf dem Geometrie Index ausgeführt werden können, muss der Geometrie Index mit Objekten gefüllt werden. Die nachfolgenden Methoden stehen dazu zur Verfügung.

Prozedur	<code>GEOINDEX.OPEN ! [[]]</code>
Beschreibung	Öffnet den Geometrie Index. Objekte mit Geometrien können in den Geometrie Index geschrieben werden.
Beispiel	<code>GEOINDEX.OPEN</code>
Prozedur	<code>GEOINDEX.WRITE_OBJECT ! [* object, g geometry][i id]</code>
Beschreibung	Schreibt ein Objekt unter einer Geometrie in den Geometrie Index. Eine eindeutige Identifikation des Objektes im Geometrie Index wird zurück gegeben. Nachdem die gewünschten Objekte in den Geometrie Index geschrieben sind, können diese mit den geometrischen Abfragen wieder gelesen werden.
Beispiel	<code>&IN IN.Geometrie GEOINDEX.WRITE_OBJECT [1]</code>
Prozedur	<code>GEOINDEX.WRITE_OBJECT_CLASS ! [s class, * object, g geometry][i id]</code>
Beschreibung	Wie <code>GEOINDEX.WRITE_OBJECT</code> aber zusätzlich kann noch eine Klasse für das Objekt mitgegeben werden.
Beispiel	<code>'MyClass' &IN IN.Geometrie GEOINDEX.WRITE_OBJECT_CLASS [1]</code>
Prozedur	<code>GEOINDEX.CLOSE ! [[]]</code>
Beschreibung	Schliesst den Geometrie Index. Der Geometrie Index wird geleert. Danach sind keine geometrischen Abfragen auf den Geometrie Index möglich.
Beispiel	<code>GEOINDEX.CLOSE</code>
Prozedur	<code>GEOINDEX.SET_CLASS_FILTER ! [s class] []</code>
Beschreibung	Für alle nachfolgenden Methoden <code>GEOINDEX.*_READ_OPEN</code> kann ein Filter gesetzt werden, welche Klassen zurückgelesen werden sollen. Die Klassen müssen vorgängig mit <code>GEOINDEX.WRITE_OBJECT_CLASS</code> in den GeoindeX geschrieben werden. Mehrere Klassen können kommasepariert gesetzt werden <code>'MyClass1,MyClass2'</code> . Ist <code>Class = NULL</code> wird der Filter zurückgesetzt, d.h. es ist kein Filter aktiv. Ein gesetzter Filter bleibt aktiv bis er wieder mit der Methode neu gesetzt oder zurückgesetzt wird.
Beispiel	<code>['MyClass'] GEOINDEX.SET_CLASS_FILTER []</code>

Alle in den Geometrie Index geschriebenen Objekte können sequentiell oder über Ihre Identifikation zurückgelesen werden. Die nachfolgenden Methoden stehen dazu zur Verfügung.

Prozedur	<code>GEOINDEX.READ_OPEN ! [] []</code>
Beschreibung	Öffnet den Geometrie Index für das sequentielle Lesen der Objekte im Geometrie Index.

Beispiel `[] GEOINDEX.READ_OPEN []`

Prozedur `GEOINDEX.READ_OBJECT ! [] [g geom, * object, i id, b status]`

Beschreibung List das nächste Objekt aus dem Geometrie Index.

Beispiel `[] GEOINDEX.READ_OBJECT [geom map 1 TRUE]`

Prozedur `GEOINDEX.READ_CLOSE ! [] []`

Beschreibung Schliesst den Geometrie Index für das sequentielle Lesen der Objekte im Geometrie Index.

Beispiel `GEOINDEX.READ_CLOSE`

Prozedur `GEOINDEX.READ_OBJECT_BY_ID ! [i id] [g geom, * object, b status]`

Beschreibung List ein Objekt aus dem Geometrie Index über dessen Identifikation.

Beispiel `[1] GEOINDEX.READ_OBJECT_BY_ID [geom map TRUE]`

Alle Objekte, deren maximale Ausdehnung ein Rechteck überlappen, können aus dem Geometrie Index gelesen werden. Die nachfolgenden Methoden stehen dazu zur Verfügung.

Prozedur `GEOINDEX.RECTANGLE_READ_OPEN ! [p point1, p point2] []`

Beschreibung Öffnet die Abfrage des Geometrie Indexes über ein Rechteck.

Beispiel `[point1 point2] GEOINDEX.RECTANGLE_READ_OPEN []`

Prozedur `GEOINDEX.RECTANGLE_READ_OBJECT ! [g geom, * object, i id, b status]`

Beschreibung Liest das nächste Objekt aus dem Geometrie Index, dessen maximale Ausdehnung das Rechteck überlappt.

Beispiel `[] GEOINDEX.RECTANGLE_READ_OBJECT [geom map 1 TRUE]`

Prozedur `GEOINDEX.RECTANGLE_READ_CLOSE ! [] []`

Beschreibung Schliesst die Abfrage.

Beispiel `[] GEOINDEX.RECTANGLE_READ_CLOSE []`

Alle Objekte, deren maximale Ausdehnung eine Fläche überlappen, können aus dem Geometrie Index gelesen werden. Die nachfolgenden Methoden stehen dazu zur Verfügung.

Prozedur `GEOINDEX.AREA_READ_OPEN ! [a area, b overlap] []`

Beschreibung Öffnet die Abfrage des Geometrie Indexes über eine Fläche. `overlap = TRUE` bedeutet: Objekte dürfen die Fläche überlappen. `overlap = FALSE` bedeutet: Objekte müssen innerhalb der Fläche sein.

Beispiel `[area TRUE] GEOINDEX.AREA_READ_OPEN []`

Prozedur `GEOINDEX.AREA_READ_OBJECT ! [g geom, * object, i id, b status]`

Beschreibung Liest das nächste Objekt aus dem Geometrie Index, dessen Geometrie die Fläche überlappt oder dessen Geometrie innerhalb der Fläche liegt (siehe `overlap`).

Beispiel `[] GEOINDEX.AREA_READ_OBJECT [geom map 1 TRUE]`

Prozedur `GEOINDEX.AREA_READ_CLOSE ! [] []`

Beschreibung Schliesst die Abfrage.

Beispiel `[] GEOINDEX.AREA_READ_CLOSE []`

Alle Objekte werden im Geometrie Index indexiert in Seiten (Pages) abgelegt. Je nach Pagsize und der maximalen Ausdehnung der Geometrie eines Objektes, wird das Objekt in einer bis mehreren Seiten indexiert. Zu der Geometrie eines Objektes können alle weiteren Objekte zurückgelesen werden, die sich in denselben Seiten des Indexes befinden. Die nachfolgenden Methoden stehen dazu zur Verfügung.

Prozedur `GEOINDEX.PAGES_READ_OPEN ! [g geometry] []`

Beschreibung Öffnet die Abfrage des Geometrie Indexes über die Ausdehnung einer Geometrie.

Beispiel `[line] GEOINDEX.PAGES_READ_OPEN []`

Prozedur `GEOINDEX.PAGES_READ_OBJECT ! [g geom, * object, i id, b status]`

Beschreibung Liest das nächste Objekt aus dem Geometrie Index, das sich in denselben Seiten des Geometrie Indexes befindet.

Beispiel `[] GEOINDEX.PAGES_READ_OBJECT [geom map 1 TRUE]`

Prozedur `GEOINDEX.PAGES_READ_CLOSE ! [] []`

Beschreibung Schliesst die Abfrage.

Beispiel `[] GEOINDEX.PAGES_CLOSE []`

Zu einem Punkt können alle Objekte mit Flächen aus dem Geometrie Index gelesen werden, bei denen ein bestimmter Punkt innerhalb der Fläche liegt. Die nachfolgenden Methoden stehen dazu zur Verfügung.

Prozedur `GEOINDEX.AREASBYPOINT_READ_OPEN ! [p point] []`

Beschreibung Öffnet die Abfrage des Geometrie Indexes über die den Punkt

Beispiel `[line] GEOINDEX.AREASBYPOINT_READ_OPEN []`

Prozedur `GEOINDEX.AREASBYPOINT_READ_OBJECT ! [g geom, * object, i id, b status]`

Beschreibung Liest das nächste Objekt mit einer Fläche aus dem Geometrie Index, in der der Punkt liegt.

Beispiel `[] GEOINDEX.AREASBYPOINT_READ_OBJECT [geom map 1 TRUE]`

Prozedur `GEOINDEX.AREASBYPOINT_READ_CLOSE ! [] []`

Beschreibung Schliesst die Abfrage.

Beispiel `[] GEOINDEX.AREASBYPOINT_CLOSE []`

Zu einer Geometrie können die nächstgelegenen Geometrien aus dem Geometrie Index gelesen werden Die nachfolgenden Methoden stehen dazu zur Verfügung.

Prozedur `GEOINDEX.NEAREST_READ_OPEN ! [g geometry] []`

Beschreibung Öffnet die Abfrage des Geometrie Indexes über die Geometrie.

Beispiel `[line] GEOINDEX.NEAREST_READ_OPEN []`

Prozedur	<code>GEOINDEX.NEAREST_READ_OBJECT ! [r distance, g geom, * object, i id, b status]</code>
Beschreibung	Liest das nächste Objekt aus dem Geometrie Index, dessen Geometrie am nächsten zur Geometrie liegt, mit der die Abfrage geöffnet wurde.
Beispiel	<code>[] GEOINDEX.AREASBYPOINT_READ_OBJECT [1.123 geom map 1 TRUE]</code>
Prozedur	<code>GEOINDEX.NEAREST_READ_CLOSE ! [] []</code>
Beschreibung	Schliesst die Abfrage.
Beispiel	<code>[] GEOINDEX.AREASBYPOINT_CLOSE []</code>

2.5. Skriptbeispiel

```
! Diese ICS Konfiguration schreibt in einem 1. Schritt
! Punkte in den Geometrie Index.
! In einem 2. Schritt werden zu einem Punkt die nächstgelegenen
! Punkte im Geometrie Index zurückgelesen.

|INCL \script\geoindex.mod

MAP GEOINDEX_PARAM
  PAGESIZE    => 50.0
  DISKMEMORY  => OFF
END_MAP

! schreiben von Punkten in den Geometrie Index
! Punkteabstand dx=10,dy=10
! von x/y-min=0/0 bis x/y-min=100/100
!-----
0.0  => VAR.ORIGIN
10.0 => VAR.DISTANCE
100.0 => VAR.LIMIT

GEOINDEX.OPEN

0  => VAR.I

VAR.ORIGIN => VAR.X
WHILE VAR.X <= VAR.LIMIT DO

  VAR.ORIGIN => VAR.Y
  WHILE VAR.Y <= VAR.LIMIT DO

    VAR.I INC => VAR.I
    IF VAR.I 100 MOD = 0 THEN
      DISPLAY VAR.I, ' objects written'
    END_IF

    VAR.I          => IN.ID
    VAR.X VAR.Y 0.0 SET_NULL TO_POINT => IN.GEOM

    &IN IN.GEOM  GEOINDEX.WRITE_OBJECT POP

    VAR.Y VAR.DISTANCE + => VAR.Y
  END_WHILE
```

```

    VAR.X VAR.DISTANCE + => VAR.X
END_WHILE

DISPLAY 'objects written=',VAR.I

! lesen der nächsten Punkte aus dem Geometrie Index
! zum Punkt 50/50
!-----

0    => VAR.I

50.0 50.0 0.0 SET_NULL TO_POINT => VAR.POINT

VAR.POINT
GEOINDEX.NEAREST_READ_OPEN
WHILE GEOINDEX.NEAREST_READ_OBJECT DO
    => VAR.ID
    => VAR.OBJECT
    => VAR.GEOM
    => VAR.DISTANCE

    VAR.I INC => VAR.I

    DISPLAY 'Distance=',VAR.DISTANCE,' from ',VAR.GEOM,' to ',VAR.POINT

END_WHILE

DISPLAY 'objects read=',VAR.I

```

3. Modul INTERSECT - Verschnitt Flächen, Linien, Punkte

3.1. Allgemeines

Mit dem Modul INTERSECT können Flächen, Linien und Punkte miteinander verschnitten werden. Zum Beispiel aus der amtlichen Vermessung die Liegenschaften mit der Bodenbedeckung und den Einzelobjekten.

INTERSECT wird mit:

```
|INCL \script\intersect.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

3.2. Abhängigkeiten von anderen Modulen

Der Modul verwendet die Klasse TOPO und GEOINDEX.

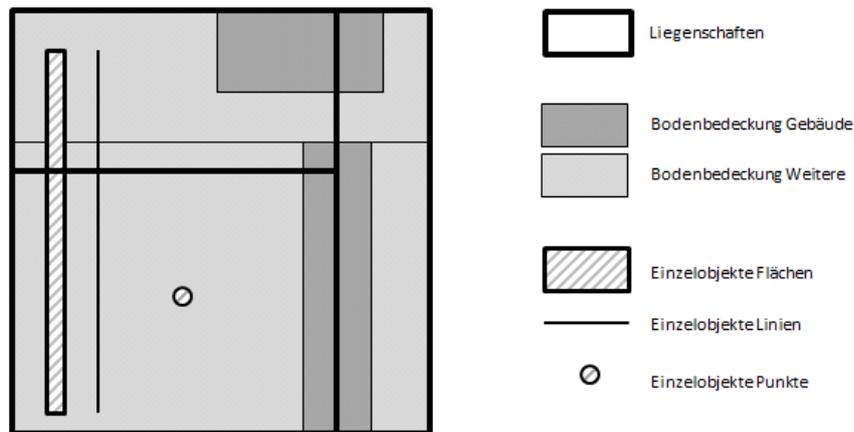


Der Modul INTERSECT kann nicht gleichzeitig mit der Klasse TOPO und/oder GEOINDEX oder einem Modul, das die Klasse TOPO und/oder GEOINDEX verwendet, angewendet werden. Bedingt die Verarbeitung einer Konfiguration eine Verwendung solcher Module mit der Klasse TOPO und/oder GEOINDEX, so ist die Konfiguration so zu gestalten, dass die Verarbeitung mit den Modulen sequentiell erfolgt

3.3. Beispiel

3.3.1. Ausgangslage

Abbildung C.1.



Das Beispiel zeigt als Ausgangslage eine Situation der Amtlichen Vermessung mit Flächen der Liegenschaften, Flächen der Bodenbedeckung und Flächen, Linien und Punkten der Einzelobjekte.

Die Objekte für einen Verschnitt werden wie folgt in das Modul INTERSECT mit der Prozedure INTERSECT_WRITE_OBJECT3, <Classname>, <Geometrie>, <Objekt-Map> geschrieben.

```
MAP INPUT_SOURCES

! INTERLIS objects read
!-----
I1 => ILTOPO,OPT.input

END_MAP

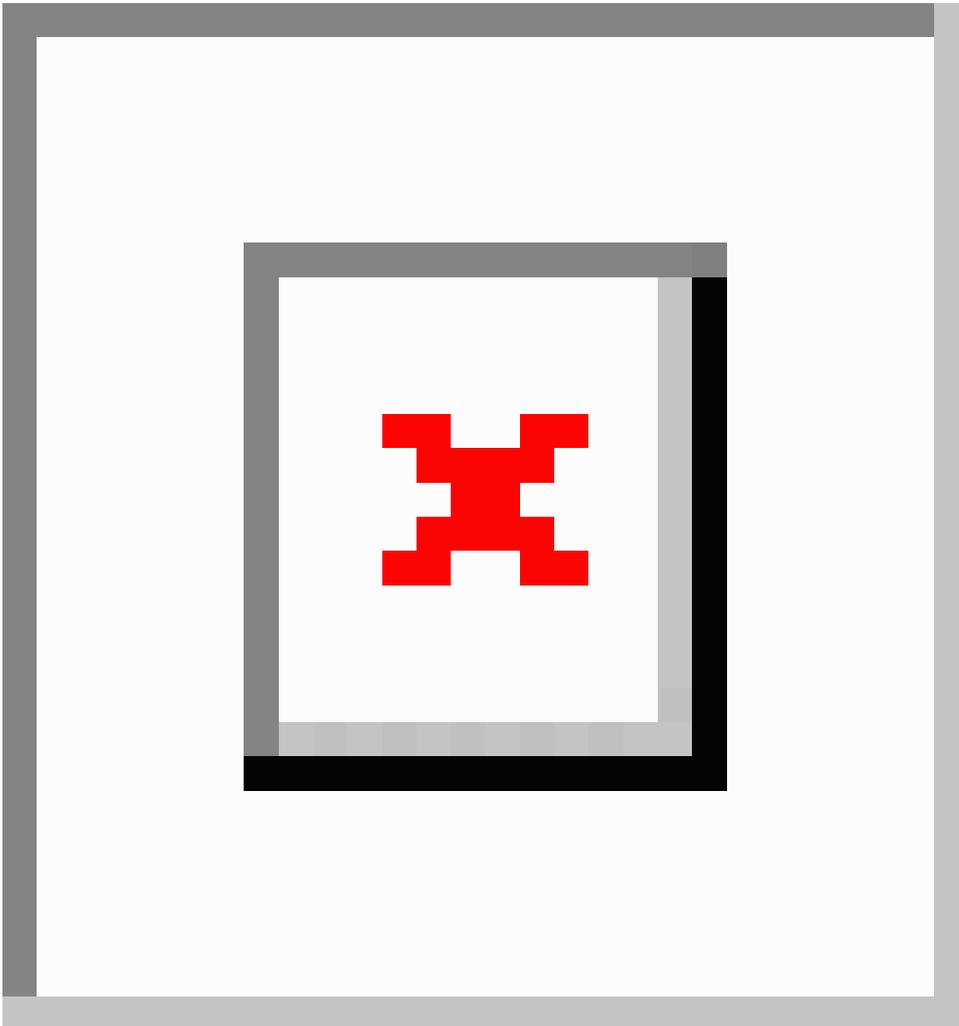
MAP INOUT

! INTERLIS objects write to intersect-module
!-----
I1                                     => IN.TOPIC,IN.TABLE
I1,Bodenbedeckung,BoFlaeche_Area      => INTERSECT_WRITE_OBJECT3,IN.TABLE,IN.GEOM,IN
I1,Einzelobjekte,Flaechenelement     => INTERSECT_WRITE_OBJECT3,EOfLaechenelement,IN.Geometrie,IN
I1,Einzelobjekte,Linienelement       => INTERSECT_WRITE_OBJECT3,EOLinienelement,IN.Geometrie,IN
I1,Einzelobjekte,Punktelement       => INTERSECT_WRITE_OBJECT3,EOPunktelement,IN.Geometrie,IN
I1,Liegenschaften,Liegenschaft_Area  => INTERSECT_WRITE_OBJECT3,IN.TABLE,IN.GEOM,IN
I1,*                                   => OFF

END_MAP
```

3.3.2. Verschnitt Modus MAININTERSECTION

Abbildung C.2.



Der Verschnitt im Modus `MAININTERSECTION` liefert pro Objekt der Hauptklasse Fläche der Liegenschaften ein Objekt. Jedes Objekt beinhaltet als Listen die Verschnittgeometrien der Unterklassen Flächen der Bodenbedeckung, Flächen der Einzelobjekte, Linien der Einzelobjekte und Punkte der Einzelobjekte.

Der Verschnitt wird wie folgt berechnet und die Objekte gelesen.

```
MAP INPUT_SOURCES

! MAININTERSECTION intersect example source
!-----
SM => INTERSECT,MAININTERSECTION,Liegenschaft_Area,BoFlaeche_Area,EOFlaechenelement,EOLinienelement,EOPunktenelement

END_MAP

MAP INOUT

! MAININTERSECTION intersect example objects read
!-----
SM => DISPLAY_INTERSECT_OBJECT0
```

END_MAP

3.3.3. Verschnitt Modus INTERSECTION

Abbildung C.3.



Der Verschnitt im Modus INTERSECTION verschneidet alle Klassen mit Flächen miteinander und liefert zu jeder Verschnittfläche ein Objekt. Jedes Objekt beinhaltet als Listen die Verschnittgeometrien der Unterklassen vom Typen Linien und Punkte.

Der Verschnitt wird wie folgt berechnet und die Objekte gelesen.

```
MAP INPUT_SOURCES

! INTERSECTION intersect example source
!-----
SI => INTERSECT,INTERSECTION,Liegenschaft_Area,BoFlaeche_Area,EOFlaechenelement,EOLinienelement,

END_MAP

MAP INOUT

! INTERSECTION intersect example objects read
!-----
SI => DISPLAY_INTERSECT_OBJECT0

END_MAP
```

3.4. Funktionsweise

Der Modul INTERSECT kennt zwei Modi von Verschnitten.

MAININTERSECTION (Verschnitt Hauptklasse mit Unterklassen)

Der Modus MAININTERSECTION verschneidet die Objekte einer Hauptklasse mit den Objekten der Unterklassen. Das heisst, jedes Objekt der Hauptklasse beinhaltet die mit der Geometrie des Hauptobjektes verschnittenen Geometrien der Unterobjekte, sofern ein Verschnitt möglich ist.

INTERSECTION (Verschnitt Klassen)

Der Modus INTERSECTION verschneidet die Objekte aller Klassen.

Beinhalten die Klassen Flächen, so werden die Flächen miteinander verschnitten und ergeben Schnittflächen. Weitere Klassen der Typen Line und Point werden mit diesen Schnittflächen verschnitten.

Beinhalten die Klassen keine Flächen, so wird die erste Klasse herangezogen und die weiteren Klassen werden mit dieser ersten Klasse verschnitten.

Die Verschneidung wird wie folgt berechnet.



Beim Modi INTERSECTION sind alle Klassen gleichberechtigt, es gibt keine Hauptklasse, respektive Hauptobjekte.



3.5. Hilfskonfiguration

Zur Anwendung des Modules stehen folgende Anteile und Konfigurationen zur Verfügung.

`ILTOOLS_DIR\system\models\INTERSECT.ili`

Allgemeines INTERLIS 1 Modell um Resultate eines Verschnittes nach INTERLIS zu transferieren.

`ILTOOLS_DIR\system\script\intersect\INTERSECT.cfg`

INTERLIS nach INTERLIS Konfiguration für einen Verschnitt der amtlichen Vermessung. Der Verschnitt wird im Logfile angezeigt und nach INTERLIS in das Modell INTERSECT.ili transferiert.

Die Konfiguration kann kopiert und einfach an andere Modelle angepasst werden.

`ILTOOLS_DIR\system\script\intersect\INTERSECT_il2shp.cfg`

INTERLIS nach Shapefile Konfiguration. Daten in INTERLIS im Modell INTERSECT.ili werden nach Shapefile transferiert.

Die Konfiguration kann kopiert und einfach an andere Modelle angepasst werden.

Dient zur Visualisierung eines Verschnittes in Shapefile.

3.6. Parametermap INTERSECT_PARAM

Folgende Parameter können in der Map INTERSECT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.
SILENT	o	ON oder OFF, Default = OFF. Meldungen anzeigen.
AREA_LIMIT	o	REAL. Default = 0.0. Flächen kleiner als die Limite nicht berücksichtigen.
STROKE	o	REAL STRING. Default = OFF. Sollen Kreisbögen in Liniensegmente aufgelöst werden. OFF : Kreisbögen werden nicht aufgelöst. REAL-Wert: definiert die Auflösung. Details siehe in der Beschreibung der Methode ICS.STROKE .
RESOLUTION	o	REAL. Default = 0.001. Geometrische Auflösung der Inputgeometrien. Für mm muss z.B. 0.001 eingegeben werden.
OVERLAP	o	REAL. Default = 0.2. Möglicher Overlap für Kreisbögen gemäss INTERLIS Spezifikation (z.B. 0.2).

3.7. Objektmodell

Dem Modul INTERSECT werden Objekte mit der vordefinierten Prozedur INTERSECT_WRITE_OBJECT3 übergeben (s.a. unten). Die Verarbeitung der Objekte erfolgt mit dem Öffnen des Modules mit der vordefinierten Prozedur INTERSECT_OPEN (s.a. unten), die mit RUN1 automatisch aufgerufen wird. Mit den Argumenten zur Prozedur INTERSECT_OPEN wird die Art der Verarbeitung und

die zu verarbeitenden Klassen definiert. Je nach Art der Verarbeitung liefert die Berechnung folgende IN-Objekte:

Komponenten für jedes IN-Objekt für die Berechnungsart `MAININTERSECTION`, `main-class`, `class`, ...

Die Berechnung im Mode `MAININTERSECTION` liefert als Objekte die originalen Objekte der Hauptklasse zurück. Zusätzlich enthält ein Hauptobjekt als Listen die Objekte der Unterklassen mit den Verschnitten, die in der Fläche des Objekt der Hauptklasse enthalten sind.

Komponente	req/opt	Beschreibung
<code>IN.INTERSECT_MODE(s)</code>	r	Verschnitt-Methode <code>INTERSECTION</code> oder <code>MAININTERSECTION</code> .
<code>IN.INTERSECT_TYPE(s)</code>	r	Geometrie Type der Verschnittgeometrie <code>area</code> , <code>line</code> oder <code>point</code> .
<code>IN.INTERSECT_GEOM(g)</code>	r	Verschnitt-Geometrie.
<code>IN.INTERSECT_AREA(r)</code>	o	Flächeninhalt wenn die Verschnitt-Geometrie eine Fläche ist.
<code>IN.INTERSECT_LENGTH(r)</code>	o	Linienlänge wenn die Verschnitt-Geometrie eine Linie ist.
<code>IN.<attribute>(*)</code>	o	Alle Attribute des originalen Objektes der Hauptklasse.
<code>IN.<class>(li)</code>	o	Eine Liste pro beteiligte Klasse, mit allen Objekten und Attributen der Klasse, die in der Verschnitt-Geometrie <code>IN.INTERSECT_GEOM</code> enthalten sind.
<code>IN.<class>.INTERSECT_MODE(s)</code>	r	Verschnitt-Methode <code>INTERSECTION</code> oder <code>MAININTERSECTION</code> .
<code>IN.<class>.INTERSECT_TYPE(s)</code>	r	Geometrie Type der Verschnittgeometrie <code>area</code> , <code>line</code> oder <code>point</code> .
<code>IN.<class>.INTERSECT_GEOM(g)</code>	r	Verschnitt-Geometrie.
<code>IN.<class>.INTERSECT_AREA(r)</code>	o	Flächeninhalt wenn die Verschnitt-Geometrie eine Fläche ist.
<code>IN.<class>.INTERSECT_LENGTH(r)</code>	o	Linienlänge wenn die Verschnitt-Geometrie eine Linie ist.
<code>IN.<class>.<attribute>(*)</code>	o	Alle Attribute des originalen Objektes.

Komponenten für jedes IN-Objekt für die Berechnungsart `INTERSECTION`, `class`, `class`, ...

Die Berechnung im Mode `INTERSECTION` liefert als Objekte die einzelnen Verschnitte der Objekte der beteiligten Klassen zurück. Neben den geometrischen Informationen der Verschnitte werden als Listen auch die in den Verschnitten enthaltenen Objekte der beteiligten Klassen zurückgegeben.

Komponente	req/opt	Beschreibung
*	r/o	Alle Komponenten wie bei der Berechnung im Mode <code>MAININTERSECTION</code> ausser den Attributen <code>IN.<attribute></code> , weil dies bei diesem Mode keine Hauptklasse gibt.

Komponenten für jedes IN-Objekt nach der Methode INTERSECT_GROUP_CLASS2, class, groupkeyattr

Nach der Berechnung des Verschnittes können beim Lesen der Verschnitt-Geometrien mit INTERSECT_GROUP_CLASS2 Unterklassen gruppiert werden. Beim Lesen der Verschnitte wird im IN-Objekt jede Unterklasse mit einer Liste der Teilgeometrie gelesen. Mit dieser Methode können die Teilgeometrien der Unterklassen nach dem Wert eines Attributes groupkeyattr gruppiert werden.

Komponente	req/opt	Beschreibung
*	r/o	Alle Attribute wie aus Verschnitt oben.
IN.INTER-SECT_AREA(r)	r	Flächeninhalt der Flächengeometrie.
IN.<attribute>(*)	o	Alle Attribute des originalen Objektes der Hauptklasse.
IN.<subclass>(li)	o	Eine Liste pro beteiligte Unterklasse, mit allen Objekten und Attributen der Unterklasse, die in der originalen Flächengeometrie IN.INTERSECT_GEOM des Objektes der Hauptklasse enthalten sind. Die Liste der Unterklasse enthält Objekte gruppiert nach den Wert des Attributes groupkeyattr.
IN.<subclass>.INTERSECT_COUNT(a)	o	Die Anzahl der Teilflächen der Unterklasse und der Gruppe.
IN.<subclass>.INTERSECT_GEOM(li)	o	Eine Liste der Teilflächen der Unterklasse und der Gruppe.
IN.<subclass>.INTERSECT_AREA(r)	o	Die Summe des Flächeninhalt der Teilflächen der Unterklasse und der Gruppe.
IN.<subclass>.<groupkeyattr>	o	Das Attribute mit dem Wert für die Gruppenermittlung der Unterklasse.
IN.<subclass>.<attribute>(*)	o	Alle Attribute des ersten gefundenen originalen Objektes der Unterklasse und der Gruppe.

3.8. Exportierte Prozeduren und Methoden

Vor der Berechnung des Verschnittes mit der Prozedur INTERSECT_OPEN müssen die Objekte des Verschnittes in das Modul geschrieben werden. Dies erfolgt mit folgender Prozedur.

Prozedur INTERSECT_WRITE_OBJECT3 ! s class, g geometry, m map

Beschreibung Schreibt eine Fläche für die Flächenberechnung in den Modul. <class> ist der Namen der Klasse des Objektes. <area> ist Flächengeometrie des Objektes. <map> ist der Namen oder die Referenz einer Map, die weitere Attribute des Objektes beinhaltet.

Beispiel

```
... => INTERSECT_WRITE_OBJECT3,Liegenschaft,IN.GEOM,IN
```

Nachdem die Objekte für den Verschnitt mit der Prozedur INTERSECT_WRITE_OBJECT3 in das Modul geschrieben worden sind, kann der Verschnitt mit der Prozedur INTERSECT_OPEN

berechnet werden. Mit der Prozedur `INTERSECT_READ_OBJECT` werden die Verschnitt-Objekte gelesen.

Prozedur `INTERSECT_OPEN ! [s type, s class, s class, ...][[]]`

Beschreibung Öffnet das Modul zwecks Berechnung des Verschnittes der definierten Klassen. Der `<type>` kann folgende Werte annehmen:

MAININTERSECTION

Die erste Klasse des Argumentes ist die Hauptklasse. Die weiteren Klassen sind Unterklassen.

Berechnet zu den Objekten der Hauptklasse die Verschnitt-Geometrien der Objekte der Unterklassen.

INTERSECTION

Berechnet zu den Objekten der definierten Klassen die Verschnitt-geometrien. Die definierten Klassen sind gleichberechtigt.

`INTERSECT_OPEN` wird von `RUN1` automatisch aufgerufen.

Die Objekte der Klassen werden mit der Prozedur `INTERSECT_WRITE_OBJECT3` für die Berechnung in das Modul geschrieben.

Beispiel `'MAININTERSECTION,Liegenschaften_Area,BoFlaeche_Area' INTERSECT_OPEN`

Prozedur `INTERSECT_READ_OBJECT ! [][b status]`

Beschreibung Liest das nächste IN-Objekt gemäss Objektmodell. `INTERSECT_READ_OBJECT` wird von `RUN1` automatisch aufgerufen.

Beispiel `INTERSECT_READ_OBJECT [TRUE]`

Prozedur `INTERSECT_CLOSE ! [][[]]`

Beschreibung Schliesst den Modul. Nach dem Schliessen kann mit `INTERSECT_OPEN` mit den im Modul enthaltenen Objekten eine neue Berechnung durchgeführt werden. `INTERSECT_CLOSE` wird von `RUN1` automatisch aufgerufen.

Beispiel `INTERSECT_CLOSE`

Prozedur `INTERSECT_GROUP_CLASS2 ! s class, s groupkeyattr`

Beschreibung Beim Lesen der Verschnittgeometrien wird im IN-Objekt jede Unterklasse mit einer Liste der Teilgeometrien zur Hauptgeometrie gelesen. Mit dieser Methode können die Teilgeometrien der Unterklassen nach dem Wert eines Attributes `groupkeyattr` gruppiert werden. Siehe dazu auch das Objektmodell.

Beispiel `... => INTERSECT_GROUP_CLASS2,Bodenbedeckung_Area,Art_TXT`

3.9. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von intersect.mod
! berechneten Objekte in der .log Datei an.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
DIALOG => FILE
```

```

MESSAGE => 'Enter .itf Input File'
FILE_FILTER => itf
FILE_EXISTS => TRUE
OPT => input
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF    => \models\DM01AVCH24D.ili
  STATISTICS      => ON
  ENUM_TO_TEXT    => ON
  CALC_SURFACE    => ON
END_MAP

MAP ILIN_TOPO
  Bodenbedeckung,BoFlaeche_Geometrie => AREA
  Liegenschaften,Liegenschaft_Geometrie => AREA
END_MAP

MAP INTERSECT_PARAM
  STATISTICS => ON
END_MAP

MAP INPUT_SOURCES

  I1 => ILTOPO,OPT.input

  S1 => INTERSECT,MAININTERSECTION,Liegenschaft_Area,BoFlaeche_Area,EOflaechenelement,EOLinienelement,EOPunktelement

END_MAP

MAP INOUT

  I1                                => IN.TOPIC,IN.TABLE
  I1,Bodenbedeckung,BoFlaeche_Area => INTERSECT_WRITE_OBJECT3,IN.TABLE,IN.GEOM,IN
  I1,Einzelobjekte,Flaechenelement => INTERSECT_WRITE_OBJECT3,EOflaechenelement,IN.Geometrie,IN
  I1,Einzelobjekte,Linienelement  => INTERSECT_WRITE_OBJECT3,EOLinienelement,IN.Geometrie,IN
  I1,Einzelobjekte,Punktelement   => INTERSECT_WRITE_OBJECT3,EOPunktelement,IN.Geometrie,IN

  I1,Liegenschaften,Liegenschaft_Area => INTERSECT_WRITE_OBJECT3,IN.TABLE,IN.GEOM,IN
  I1,*                                => OFF

  S1 => DISPLAY_OBJECT1,IN

END_MAP

MAP MACRO
END_MAP

|INCL \script\iltopo.mod
|INCL \script\intersect.mod
|INCL \script\run1.prg

```

4. Modul LIST - ICS Objekte temporär speichern

4.1. Allgemeines

Mit dem Modul LIST können ICS Objekte im Hauptspeicher zwischengespeichert und aus dem Zwischenspeicher während der Verarbeitung sequentiell wieder ausgelesen werden. Der Modul hat eine sehr ähnliche Funktionalität wie der Modul OSTREAM. Der Unterschied zu OSTREAM besteht darin, dass der Zwischenspeicher im Fall von OSTREAM Dateien sind und im Fall von LIST der Hauptspeicher. LIST eignet sich daher nur für kleinere Datenmengen ist dafür aber schneller als OSTREAM.

LIST wird mit:

```
|INCL \script\list.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

4.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

4.3. Parametermap

Keine Parametermap notwendig.

4.4. Objektmodell

Der Modul LIST speichert bzw. liefert die ICS Objekte so wie die Objekte ihm ursprünglich übergeben wurden. Der Modul LIST hat daher kein eigenes Objektmodell.

4.5. Exportierte Prozeduren und Methoden

Prozedur LIST_OPEN ! [s list][[]]

Beschreibung Öffnet die Liste mit Namen <list>.

Beispiel 'lfp' LIST_OPEN

Prozedur LIST_READ_OBJECT ! [[]b state]

Beschreibung Liest das nächste Objekt aus der aktuellen Liste.

Beispiel LIST_READ_OBJECT [TRUE]

Prozedur LIST_CLOSE ! [[]]

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel LIST_CLOSE

Prozedur LIST_WRITE_OBJECT2 ! s list,m map

Beschreibung Schreibt die Map <map> in die Liste <list>.

Beispiel ... => LIST_WRITE_OBJECT2,lfp,IN

4.6. Skriptbeispiel

```

! Diese ICS Konfiguration speichert alle LFP Fixpunkte aus
! einer INTERLIS Datei in einer Liste. Am Schluss des Skripts
! werden die Nummern der in der Liste enthaltenen Punkte in
! die .log Datei ausgegeben.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\Grunddatensatz.ili
  STATISTICS   => ON
  DEBUG        => OFF
END_MAP

MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
  I2 => LIST, lfp
END_MAP

MAP INOUT
  I1 => IN.TOPIC, IN.TABLE
  I1,Fixpunkte,LFP => LIST_WRITE_OBJECT2, lfp, IN
  I1,* => OFF
  I2 => DISPLAY_OBJECT1, IN.Nummer
END_MAP

|INCL \script\util.lib
|INCL \script\list.mod
|INCL \script\ilin.mod
|INCL \script\run1.prg

```

5. Modul MAP - ICS Objekte temporär speichern

5.1. Allgemeines

Mit dem Modul MAP können ICS Objekte im Hauptspeicher zwischengespeichert und aus dem Zwischenspeicher während der Verarbeitung wieder ausgelesen werden. Der Modul unterscheidet sich von LIST vor allem durch die Möglichkeit jedes Objekt unter einem eindeutigen Schlüssel speichern und später wieder abfragen zu können.

MAP wird mit:

```
|INCL \script\map.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

5.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

5.3. Parametermap

Der Modul MAP benötigt für das Speichern der Objekte benannte Map's. Diese müssen vom Benutzer in der .cfg Datei vorgängig angelegt werden.

5.4. Objektmodell

Komponenten für jedes IN-Objekt

Komponente	req/opt	Beschreibung
IN.KEY(s)	r	Schlüssel welcher beim Speichern des Werts in der Map übergeben wurde.
IN.VALUE(o)	r	Wert welcher unter Schlüssel IN.KEY in der Map abgelegt ist.

5.5. Exportierte Prozeduren und Methoden

Prozedur `MAP_OPEN ! [s map][]`

Beschreibung Öffnet die Map mit Namen <map>.

Beispiel `'lfp' MAP_OPEN`

Prozedur `MAP_READ_OBJECT ! [][b state]`

Beschreibung Liest das nächste Objekt aus der aktuellen Map.

Beispiel `MAP_READ_OBJECT [TRUE]`

Prozedur `MAP_CLOSE ! [][]`

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel `MAP_CLOSE`

Prozedur `MAP_WRITE_OBJECT3 ! s map,s key,o value`

Beschreibung Schreibt das Objekt <value> in die Map <map> unter dem Schlüssel <key>.

Beispiel `... => MAP_WRITE_OBJECT3,lfp,IN.Nummer,IN`

5.6. Skriptbeispiel

```
! Diese ICS Konfiguration speichert alle LFP Fixpunkte aus
! einer INTERLIS Datei unter ihrer Nummer in der Map LFP.
! Am Schluss des Skripts werden die Nummern der in der Map
! enthaltenen Punkte in die .log Datei ausgegeben. Durch
! das Speichern in der Map LFP werden doppelte Nummern
! automatisch eliminiert.
```

```

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\Grunddatensatz.ili
  STATISTICS => ON
  DEBUG => OFF
END_MAP

MAP LFP
END_MAP

MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
  I2 => MAP,LFP
END_MAP

MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => MAP_WRITE_OBJECT3,LFP,IN.Nummer,IN
  I1,* => OFF
  I2 => DISPLAY_OBJECT1,IN.KEY
END_MAP

|INCL \script\util.lib
|INCL \script\map.mod
|INCL \script\ilin.mod
|INCL \script\run1.prg

```

6. Modul MTOPO - Mehrere AREA Topologien berechnen

6.1. Allgemeines

Mit dem Modul MTOPO können geschlossene Flächen aus Begrenzungslinien (Boundaries) und Zentroiden berechnet werden. Im Gegensatz zum Modul TOPO können mit MTOPO mehrere Flächennetze gleichzeitig berechnet werden.

MTOPO wird mit:

```
|INCL \script\mtopo.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

6.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

6.3. Parametermap TOPO_PARAM

Folgende Parameter können in der Map TOPO_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
RESOLUTION	r	Geometrische Auflösung der Inputgeometrien. Für mm muss z.B. 0.001 eingegeben werden.
OVERLAP	r	Möglicher Overlap für Kreisbögen gemäss INTERLIS Spezifikation (z.B. 0.2).



Falls die Map TOPO_PARAM nicht definiert wird, wird die Map TOPO_PARAM vom Modul MTOPO automatisch mit Standardwerten erzeugt.

6.4. Objektmodell

Dem Modul MTOPO werden Objekte mit den vordefinierten Prozeduren TOPO_WRITE_CENTROID3 bzw. TOPO_WRITE_BOUNDARY2 übergeben (s.a. unten). Nach der Toplogieberechnung liefert der MTOPO Modul folgende IN-Objekte:

Allgemeine Komponenten für jedes IN-Objekt

Komponente	req/opt	Beschreibung
IN.LAYER(s)	r	Layer des gelieferten Objekts (s.a. unten).
IN.GEOM(a)	r	Geometrie der berechneten Fläche.
IN.VALUE(s)	r	Attributwert der berechneten Fläche.

6.5. Exportierte Prozeduren und Methoden

Prozedur MTOPO_OPEN ! [s arg][[]]

Beschreibung Öffnet den Toplogiemodul. Das Argument <arg> hat im Moment keine Bedeutung. MTOPO_OPEN wird von RUN1 automatisch aufgerufen.

Beispiel `' ' MTOPO_OPEN`

Prozedur MTOPO_READ_OBJECT ! [][b status]

Beschreibung Liest das nächste IN-Objekt gemäss Objektmodell. MTOPO_READ_OBJECT wird von RUN1 automatisch aufgerufen.

Beispiel `MTOPO_READ_OBJECT [TRUE]`

Prozedur MTOPO_CLOSE ! [][[]]

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Alle temporären Dateien werden wieder gelöscht. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel `MTOPO_CLOSE`

Prozedur MTOPO_WRITE_BOUNDARY2 ! s layer, l boundary

Beschreibung Schreibt die Begrenzungslinie <boundary> in den Topologiebuffer <layer>. Es können beliebig viele Topologiebuffer via <layer> angesprochen werden.

Beispiel `... => MTOPO_WRITE_BOUNDARY2,buffer1,IN.GEOM`

Prozedur	<code>MTOPO_WRITE_CENTROID3 ! s layer, o value, p centroidpoint</code>
Beschreibung	Schreibt das Zentroid in den Topologiebuffer <layer>. Das Zentroid erhält den Attributwert <value> und der Zentroidpunkt ist <centroidpoint>.
Beispiel	<code>... => MTOPO_WRITE_CENTROID3,buffer1,IN.TXT,IN.GEOM</code>

 Es ist auch möglich die Topologie mit der Klasse TOPO ohne die Verwendung von RUN1 zu berechnen (s.a. iG/Script Benutzer- und Referenzhandbuch).

6.6. Skriptbeispiel

```
! Diese ICS Konfiugration berechnet aus DXF Polylines geschlossene
! Flaechen. Die Flaechen werden in der .log Datei angezeigt.
! Fuer die Berechnung wird der MTOPO Modul eingesetzt.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .dxf Input File'
  FILE_FILTER => dxf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP
```

```
MAP DXFIN_PARAM
  STATISTICS => ON
  DEBUG => OFF
END_MAP
```

```
MAP INPUT_SOURCES
  I1 => DXFIN,OPT.input
  I2 => MTOPO
END_MAP
```

```
MAP INOUT
  I1 => IN.LAYER
  I1,01234 => MTOPO_WRITE_BOUNDARY2,buffer1,IN.GEOM
  I1,01235 => MTOPO_WRITE_CENTROID3,buffer1,IN.TXT,IN.GEOM
  I1,* => OFF
  I2 => DISPLAY_OBJECT1,IN
END_MAP
```

```
|INCL \script\util.lib
|INCL \script\mtopo.mod
|INCL \script\dxfin.mod
|INCL \script\run1.prg
```

7. Modul NOOP - Spezielle Initialisierungen

7.1. Allgemeines

Mit dem Modul NOOP (= no operation) kann in den Ablauf eines Konfigurationsscripts eingegriffen werden. NOOP ist z.B. nützlich, um Initialisierungen von Benutzermaps, Variablen, etc. vorzunehmen. NOOP steht alternativ zu den Triggerprozeduren der Inputquellen zur Verfügung. Manchmal können Initialisierungen mit NOOP übersichtlicher konfiguriert werden als mit Triggerprozeduren.

NOOP wird mit:

```
|INCL \script\noop.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

7.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

7.3. Parametermap

Keine Parametermap notwendig.

7.4. Objektmodell

NOOP liefert ein einziges leeres IN Objekt. NOOP verfügt daher über kein eigenes Objektmodell.

7.5. Exportierte Prozeduren und Methoden

NOOP exportiert keine Prozeduren oder Methoden. Alle von NOOP implementierten Prozeduren werden von RUN1 automatisch aufgerufen. Einer mit NOOP definierten Inputquelle kann als Argument eine Meldung übergeben werden, welche beim Ausführen von NOOP in die .log Datei ausgegeben wird.

7.6. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt für jedes von  
! SHPIN gelesene Objekt die Meldung 'hello, world!' an.  
! NOOP wird fuer die Initialisierung der Benutzervariable  
! VAR.MESSAGE benutzt.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1  
  DIALOG => FILE ! FILE | STRING | ODBC  
  MESSAGE => 'Enter .shp Input File'  
  FILE_FILTER => shp  
  FILE_EXISTS => TRUE  
  OPT => input  
END_MAP
```

```

MAP SHPIN_PARAM
  STATISTICS => ON
  DEBUG      => OFF
END_MAP

MAP INPUT_SOURCES
  I0 => NOOP,'Variablen Initialisierung ...'
  I1 => SHPIN,OPT.input
END_MAP

MAP INOUT
  I0 => INIT_MYVAR0
  I1 => DISPLAY_OBJECT1,VAR.MESSAGE
END_MAP

|INCL \script\util.lib
|INCL \script\noop.mod
|INCL \script\shpin.mod

PROCEDURE INIT_MYVAR0
  'hello, world!' => VAR.MESSAGE
END_PROCEDURE

|INCL \script\run1.prg

```

8. Modul OSTREAM - ICS Objekte temporär speichern

8.1. Allgemeines

Mit dem Modul OSTREAM (= Objektstrom) können ICS Objekte in temporären Dateien zwischengespeichert und aus dem Zwischenspeicher während der Verarbeitung wieder sequentiell oder über einen Schlüssel ausgelesen werden. Der Modul hat eine sehr ähnliche Funktionalität wie der Modul LIST. Der Unterschied zu LIST besteht darin, dass der Zwischenspeicher im Fall von OSTREAM Dateien sind und im Fall von LIST der Hauptspeicher. OSTREAM eignet sich daher für grosse Datenmengen ist dafür aber langsamer als LIST.

OSTREAM wird mit:

```
|INCL \script\ostream.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

8.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

8.3. Parametermap

Keine Parametermap notwendig.

8.4. Objektmodell

Der Modul OSTREAM speichert bzw. liefert die ICS Objekte so wie die Objekte ihm ursprünglich übergeben wurden. Der Modul OSTREAM hat daher kein eigenes Objektmodell.

8.5. Exportierte Prozeduren und Methoden

Für eine sequentielle Verarbeitung von Objekten.

Prozedur OSTREAM_OPEN ! [s file][]
Beschreibung Öffnet einen Objektstrom mit Namen <file>. <file> wird immer relativ zu OPT.temp_dir angelegt.

Beispiel 'test.dat' OSTREAM_OPEN

Prozedur OSTREAM_READ_OBJECT ! [[b status]
Beschreibung Liest das nächste Objekt aus dem aktuellen Objektstrom.

Beispiel OSTREAM_READ_OBJECT [TRUE]

Prozedur OSTREAM_CLOSE ! [[]]
Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Alle temporären Dateien werden wieder gelöscht. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel OSTREAM_CLOSE

Prozedur OSTREAM_WRITE_OBJECT1 ! s file
Beschreibung Schreibt das aktuelle IN-Objekt in die temporäre Datei <file>.

Beispiel ... => OSTREAM_WRITE_OBJECT1,test.dat

Für eine Verarbeitung von Objekten über Schlüssel.

Prozedur OSTREAM_WRITE_OBJECT_KEY3 ! s file, s key, m object
Beschreibung Schreibt das Objekt <m> mit dem Schlüssel <key> in die temporäre Datei <file>.

Beispiel ... => OSTREAM_WRITE_OBJECT_KEY3,test.dat,IN.OBJID,IN

Prozedur OSTREAM_WRITE_OBJECT_KEY [s file, s key, m map][]
Beschreibung Schreibt das Objekt <m> mit dem Schlüssel <key> in die temporäre Datei <file>.

Beispiel 'test.dat' IN.OBJID &IN OSTREAM_WRITE_OBJECT_KEY

Prozedur OSTREAM_READ_OBJECT_KEY [s file, s key][m map, b status]
Beschreibung Liest das Objekt <m> mit dem Schlüssel <key> aus der temporäre Datei <file>. Das Objekt <m> wird nur auf dem Stack geliefert, wenn der status = TRUE ist.

Beispiel 'test.dat' '10001' OSTREAM_READ_OBJECT_KEY [m TRUE]

8.6. Skriptbeispiel

```

! Diese ICS Konfiguration speichert alle LFP Fixpunkte aus
! einer INTERLIS Datei in einer temporaeren Datei. Am Schluss des
! Skripts werden die Nummern der in der temporaeren Datei enthaltenen
! Punkte in die .log Datei ausgegeben.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\Grunddatensatz.ili
  STATISTICS   => ON
  DEBUG        => OFF
END_MAP

MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
  I2 => OSTREAM,lfp.dat
END_MAP

MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,Fixpunkte,LFP => OSTREAM_WRITE_OBJECT1,lfp.dat
  I1,* => OFF
  I2 => DISPLAY_OBJECT1,IN.Nummer
END_MAP

|INCL \script\util.lib
|INCL \script\ostream.mod
|INCL \script\ilin.mod
|INCL \script\run1.prg

```

9. Modul PLOT - Plotlayout schreiben

9.1. Allgemeines

Mit dem Modul PLOT können Plotlayouts mit Elementen wie Titelblatt, Nordpfeil, Koordinatenkreuze, etc. erzeugt werden. Der Modul Plot wird typischerweise in Kombination mit dem Modul PSOUT für PostScript/PDF,JPG/TIFF-Dateien oder DXFOUT für DXF/DWG-Dateine eingesetzt.

Der Modul wird mit:

```
|INCL \script\plot.mod
```

in einer ICS Konfiguration verfügbar gemacht.

9.2. Abhängigkeiten von anderen Modulen

Der Modul wird nicht als selbständiges Modul, sondern als Zusatz zu anderen Modulen, z.B. PSOUT oder DXFOUT, eingesetzt.

9.3. Parametermap PLOT_PARAM

Folgende Parameter können in der Map PLOT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
TYPE	r	PLOT RASTER OFF. Definiert den Plottyp. Der Plottyp PLOT basiert auf einem Format A0 bis A4. Der Plottyp RASTER erzeugt ein Rasterfile und wird im Zusammenhang mit dem Modul PSOUT verwendet. Der Wert OFF ignoriert die Plotanteile.
FORMAT	o	A0 A1 A2 A3 A4. Definiert das Plot-Papierformat.
FORMATORIENTATION	o	hoch quer. Definiert die Ausrichtung des Plot-Papierformates.
ORIGIN	o	<x>/<y>. Definiert den Ursprung des Plotes.
ANGLE	o	<r>. Definiert den Drehwinkel des Plotes
SCALE	o	1:<scale>. Definiert den Massstab für des Plotes
WIDTH	o	<m>. Definiert die Breite im Massstab 1:1 der Daten in Meter es Plotes
HEIGHT	o	<m>. Definiert die Höhe im Massstab 1:1 der Daten in Meter es Plotes
LAYOUT	o	ON OFF. Definiert, ob die Layout-Element des Plotes erzeugt werden.
COORDCROSS	o	ON OFF. Definiert, ob die Koordinatenkreuze des Plotes erzeugt werden.
COORDCROSS	o	ON OFF. Definiert, ob die Koordinatenband des Plotes erzeugt werden.
SCALEBAND	o	ON OFF. Definiert, ob die Skalierungband des Plotes erzeugt werden.
DATA_URL	o	<string>. Definiert ein GeoShop-URL für ein Bild. Beispiel: http://localhost:3501/image?reqid=image1 &view=av &layers=AV_Bodenbedeckung,AV_Liegenschaften,AV_Fixpunkte,AV_Einzelobjekte &dpi=100&user=test&password=test
ADJUST	o	ON OFF. Default = OFF. Definiert ob alle Input-Daten in den Plot eingepasst werden sollen.

Mit den aufgeführten Parameter lassen sich folgende Typen von Outputs generieren.

Output Type Plot

Dieser Typ benötigt immer einen Plotlayout. Dazu mehr im nächsten Kapitel. Folgende Parameter werden für diesen Plottyp benötigt

Required immer

PLOT_PARAM.TYPE='PLOT'

PLOT_PARAM.FORMAT

Papiergrösse

PLOT_PARAM.FORMATORIENTATION

Papierausrichtung

Required 1

Folgende Parameter setzen, falls der Plot einen bestimmten Ausschnitt und einen bestimmten Massstab besitzt.

PLOT_PARAM.ORIGIN	x/y-Koordinate Ausschnitt unten links
PLOT_PARAM.ANGLE	Rotation Ausschnitt
PLOT_PARAM.SCALE	Massstab

Required 2

Folgende Parameter/Optionen setzen, falls Daten in den Plot eingepasst werden sollen.

PLOT_PARAM.ADJUST=ON	ON: Die Daten werden in das Format eingepasst PLOT_PARAM.SCALE wird nicht berücksichtigt
----------------------	--

PLOT_PARAM.ORIGIN	x/y-Koordinate Ausschnitt unten links
PLOT_PARAM.ANGLE	Rotation Ausschnitt
PLOT_PARAM.WIDTH	Data Breite in Meter
PLOT_PARAM.HEIGHT	Data Höhe in Meter

Falls diese Parameter nicht gesetzt sind, werden alle Daten eingepasst, sonst nur der spezifizierte Bereich.

Output Typ Raster

Typischerweise betrifft dies ein tif-Output. Andere Fileformate wie pdf/jpg werden aber auch unterstützt. Folgende Parameter werden für diesen Plottyp benötigt

Required

PLOT_PARAM.TYPE='RASTER'	
PLOT_PARAM.ORIGIN	x/y-Koordinate Ausschnitt unten links
PLOT_PARAM.ANGLE	Rotation Ausschnitt
PLOT_PARAM.WIDTH	Data Breite in Meter
PLOT_PARAM.HEIGHT	Data Höhe in Meter

Optionen

Alle Parameter lassen sich von aussen mit Optionen übersteuern. Für den Namen der Option für den entsprechenden Parameter gilt:

PLOT_PARAM.<parameter> entspricht OPT.plot_<parameter>

wobei in der Option der Parameter klein geschrieben wird.

Beispiele:

PLOT_PARAMETER.TYPE entspricht OPT.plot_type

PLOT_PARAMETER.FORMAT entspricht OPT.plot_format

So lassen sich die Parameter als Optionen an eine Konfiguration übergeben:

```
ICS_DIR\system\bin\ics.opt -script mypdf.cfg -plot_type PLOT -plot_format A4 ...
```

9.4. Plotlayout Map PLOT_LAYOUT

Für den Output Typ PLOT mit PLOT_PARAM.TYPE => PLOT werden Plotlayouts benötigt, die den Layout des Plots definieren. Ein Plotlayout enthält Definitionen wie den Titel oder den Nordpfeil für einen Plot. Welcher Plotlayout für welches Papierformat und Massstab verwendet werden soll, wird in der Map PLOT_LAYOUT definiert. Diese Map beinhaltet folgende Definitionen:

```
MAP PLOT_LAYOUT
```

```
...
```

```
<format>,<formatorientation>,<scale> => <Layout-INTERLIS-File>
```

```
...
```

```
END_MAP
```

```
<format>
```

Definiert das Papierformat für den Plotlayout. Papierformate sind A0,A1,A2,A3,A4 oder * wenn der Plotlayout für alle Papierformate gilt.

```
<formatorientation>
```

Definiert die Ausrichtung des Papierformates für den Plotlayout. Ausrichtungen sind quer, hoch oder * wenn der Plotlayout für alle Ausrichtungen gilt.

```
<scale>
```

Definiert den Massstab für den Plotlayout. Massstäbe sind beliebige im Format 1:<Massstab> * wenn der Plotlayout für alle Massstäbe gilt.

```
<Layout-INTERLIS-File>
```

Definiert die INTERLIS Datei welche die Plotlayout-Elemente enthält.

Beispiel einer Definition:

```
MAP PLOT_LAYOUT
! Format, hoch|quer, Massstab
A0, quer, * => \plot\a0q500.itf
A0, hoch, * => \plot\a0h500.itf
A1, quer, * => \plot\a1q500.itf
A1, hoch, * => \plot\a1h500.itf
A2, quer, * => \plot\a2q500.itf
A2, hoch, * => \plot\a2h500.itf
A3, quer, * => \plot\a3q500.itf
A3, hoch, * => \plot\a3h500.itf
A4, quer, * => \plot\a4q500.itf
A4, hoch, * => \plot\a4h500.itf
DEFAULT => \plot\a4h500.itf
END_MAP
```

Die Daten der INTERLIS Files mit den Plotlayout-Elementen müssen dem INTERLIS Modell PlotLayout.ili entsprechen. Dieses Modell ist unter ILTOOLS_DIR\system\plot\PlotLayout.ili vorhanden. Im demselben Verzeichnis sind auch INTERLIS Beispieldateien für verschiedene Plotlayouts vorhanden.

Wie Plotlyouts erstellt werden, ist im folgenden Dokument beschrieben.

ICS Plotlayouts

9.5. Koordinatenkreuze Map PLOT_COORDCROSS_WIDTH

In dieser Map kann für die Koordinatenkreuze und das Koordinatenband der Abstand der Koordinatenkreuze pro Massstab definiert werden.

```
MAP PLOT_COORDCROSS_WIDTH
...
<scale> => <distance>
...
END_MAP
```

<scale>

Definiert den Plotmassstab.

<distance>

Definiert den Abstand der Koordinatenkreuze.

Beispiel einer Definition:

```
MAP PLOT_COORDCROSS_WIDTH
'1:250'      => 50.0
'1:500'      => 50.0
'1:1000'     => 100.0
DEFAULT     => 100.0
END_MAP
```

9.6. Skalierungsband Map PLOT_SCALEBAND_WIDTH

In dieser Map kann für das Skalierungsband Breite,Höhe,Offset und Texthöhe pro Massstab definiert werden.

```
MAP PLOT_COORDCROSS_WIDTH
```

```
...
<scale> => <width>[,<height>,<offsetx>,<offsety>,<textheight>]
```

```
...
```

```
END_MAP
```

<scale>

Definiert den Plotmassstab.

<width>

Definiert die Breite des Skalierungsbandes.

<height>

Definiert die Höhe des Skalierungsbandes.

<offsetx>

Definiert den X-Offset Plotlayout-Border unten links zu Skalierungsband unten links

<offsety>

Definiert den Y-Offset Plotlayout-Border unten links zu Skalierungsband unten links

<textheight>

Definiert die Texthöhe der Anschrift des Skalierungsband

Beispiel einer Definition:

```
MAP PLOT_SCALEBAND_WIDTH
'1:250'      => 12.0
'1:500'      => 20.0
'1:1000'     => 40.0
END_MAP
```

9.7. Werte Map PLOT_VALUES

In dieser Map können Werte für den Plot gesetzt werden. In der Regel handelt es sich bei den Werten um Platzhalter die im Plotlayout verwendet werden, wie den Massstab oder Werte für das Titelblatt.

```
MAP PLOT_VALUES
```

```
...
<object> => <variable for value>
```

```
...
```

```
END_MAP
```

<object>

Ein Objekt kann eine Variable, eine Prozedur oder eine Konstante sein. Das Objekt liefert den Wert.

<variable for value>

Definiert die Variable, die den Wert enthalten soll.

Beispiel einer Definition:

```
MAP PLOT_VALUES
  PLOT_PARAM.SCALE => OPT.massstab
  PLOT_DATE        => OPT.datum
  'Demogemeinde'  => OPT.gemeinde
END_MAP
```

9.8. Objekt Map PLOT_WRITE_OBJECT

In dieser Map wird definiert, wie die Plot-Objekte geschrieben werden. Das schreiben ist abhängig vom Output-Modul und entspricht dem gängigen Verfahren wie mir RUN1.

```
MAP PLOT_WRITE_OBJECT
```

```
...
<topic>,<table> => <rule to write>
```

```
...
```

```
END_MAP
```

<topic>,<table>

Topic und Table des Plot-Objektes entsprechend dem Modell PlotLayout.ili

<rule to write>

Schreibregel entsprechend dem Output-Module.

Beispiel einer Definition (Output-Modul PSOUT):

```
MAP PLOT_WRITE_OBJECT
  Plot_Elemente,Border_Flaeche      => OFF
  Plot_Elemente,Clipp_Flaeche       => PSOUT_WRITE_CLIP2,IN.Geometrie,-1001
  Plot_Elemente,Flaeche             => PSOUT_WRITE_POLYGON3,IN.Geometrie,white,-1005
  Plot_Elemente,Linie               => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous
  Plot_Elemente,Bild                => PSOUT_WRITE_JPG6,IN.Geometrie,IN.Ori,IN.Width
  Plot_Elemente,Text                => PSOUT_WRITE_TEXT12,IN.Text,IN.Geometrie,IN.Or
  Plot_Elemente,Symbol_Linie        => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous
  Plot_Elemente,Symbol_Flaeche      => PSOUT_WRITE_POLYGON3,IN.Geometrie,black,-1005

  Plot_Generiert,Koordinatenkreuz   => PSOUT_WRITE_SYMBOL7,IN.Geometrie,IN.Ori,09705
  Plot_Generiert,Koordinatenband_Linie => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous
  Plot_Generiert,Koordinatenband_Text => PSOUT_WRITE_TEXT12,IN.Text,IN.Geometrie,IN.Or
  Plot_Generiert,Skalierungsband_Clipp_Flaeche => PSOUT_WRITE_POLYGON3,IN.Geometrie,white,10000
```

```

Plot_Generiert,Skalierungsband_Border_Linie => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous,blac
Plot_Generiert,Skalierungsband_Block_Flaeche1 => PSOUT_WRITE_POLYLINE5,IN.Geometrie,continuous,blac
Plot_Generiert,Skalierungsband_Block_Flaeche2 => PSOUT_WRITE_POLYGON3,IN.Geometrie,black,100001
Plot_Generiert,Skalierungsband_Text => PSOUT_WRITE_TEXT12,IN.Text,IN.Geometrie,IN.Ori,IN.
END_MAP

```

9.9. Anwendung

Das Modul kann auf zwei Arten angewendet werden.

Anwendung nicht als Input Source.

Das Modul wird inkludiert, die Parameter werden gesetzt. Nach der Verarbeitung der eigentlichen Daten werden die Plot-Objekte durch das Module automatisch verarbeitet.

Anwendung als Input Source.

Das Modul wird inkludiert. Das Modul wird als Input-Source definiert. Die Plot-Elemente werden mit RUN1 als Objekte geliefert und müssen verarbeiten werden.

9.10. Exportierte Prozeduren und Methoden

Zur Anwendung mit RUN1 mit dem Plot-Mould als Input Source.

Prozedur PLOT_OPEN ! [[]]

Beschreibung Öffnet aufgrund der Plotparameter Format, Formatorientation und Massstab das entsprechende Plotlayout und initialisiert den Modul. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel PLOT_OPEN

Prozedur PLOT_READ_OBJECT [[]b state]

Beschreibung Liest das nächste IN-Objekt aus der geöffneten Plotlayouts. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel PLOT_READ_OBJECT [TRUE]

Prozedur PLOT_CLOSE ! [[]]

Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.

Beispiel PLOT_CLOSE

9.11. Skriptbeispiel

```

! Anwendung plot.mod nicht als Input Source.
! Diese ICS Konfiguration schreibt von ilin.mod
! gelesenen Objekte mit dxfout.mod in eine DXF-Datei
! und schreibt zusätzlich die Plotlayout-Objekte

```

```
|LICENSE \license\iltools.lic
```

```

MAP USER_INPUT1
  DIALOG => FILE
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf

```

```
FILE_EXISTS => TRUE
OPT => input
END_MAP

MAP USER_INPUT2
DIALOG => FILE
MESSAGE => 'Enter .dxf Output File'
FILE_FILTER => dxf
FILE_EXISTS => FALSE
OPT => output
END_MAP

MAP ILIN_PARAM
INTERLIS_DEF => \models\dm01avch24d.ili
LOG_TABLE => ON
TRACE => OFF
STATISTICS => ON
CALC_SURFACE => ON
END_MAP

MAP DXFOUT_PARAM
TEMPLATE => \data\geobau2.tem
CREATE_LAYERS => ON
STATISTICS => ON
END_MAP

MAP PLOT_PARAM
TYPE           => PLOT
FORMAT         => A4
FORMATORIENTATION => hoch
ORIGIN         => '675855/245385'
ANGLE          => '45.0'
SCALE          => '1:500'
LAYOUT         => ON
COORDCROSS    => ON
COORDBAND     => ON
SCALEBAND     => ON
END_MAP

MAP PLOT_LAYOUT
A0,quer,*     => \plot\a0q500.itf
A0,hoch,*    => \plot\a0h500.itf
A1,quer,*     => \plot\a1q500.itf
A1,hoch,*    => \plot\a1h500.itf
A2,quer,*     => \plot\a2q500.itf
A2,hoch,*    => \plot\a2h500.itf
A3,quer,*     => \plot\a3q500.itf
A3,hoch,*    => \plot\a3h500.itf
A4,quer,*     => \plot\a4q500.itf
A4,hoch,*    => \plot\a4h500.itf
DEFAULT      => \plot\a4h500.itf
END_MAP

MAP PLOT_COORDCROSS_WIDTH
'1:250'      => 100.0
'1:500'      => 100.0
'1:1000'     => 100.0
DEFAULT      => 100.0
```

```

END_MAP

MAP PLOT_SCALEBAND_WIDTH
  '1:250'      => 12.0
  '1:500'      => 20.0
  '1:1000'     => 40.0
END_MAP

MAP PLOT_VALUES
  PLOT_PARAM.SCALE => OPT.massstab
  PLOT_DATE         => OPT.datum
  'Demogemeinde'   => OPT.gemeinde
END_MAP

MAP PLOT_WRITE_OBJECT
  Plot_Elemente,Border_Flaeche      => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,plot,CONTINUOUS
  Plot_Elemente,Clipp_Flaeche       => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,plot_clipp,CONTINUOUS
  Plot_Elemente,Flaeche              => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,plot,CONTINUOUS
  Plot_Elemente,Linie                => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,plot,CONTINUOUS
  Plot_Elemente,Bild                 => PSOUT_WRITE_JPG6,IN.Geometrie,IN.Ori,IN.Width,IN.Height
  Plot_Elemente,Text                => DXFOUT_WRITE_TEXT6,IN.Text,IN.Geometrie,IN.Ori,IN.Width,IN.Height
  Plot_Elemente,Symbol_Linie        => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,plot,CONTINUOUS
  Plot_Elemente,Symbol_Flaeche      => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,plot,CONTINUOUS

  Plot_Generiert,Koordinatenkreuz   => DXFOUT_WRITE_BLOCK3,IN.Geometrie,0.0,coordcross,CONTINUOUS
  Plot_Generiert,Koordinatenband_Linie => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,coordband,CONTINUOUS
  Plot_Generiert,Koordinatenband_Text => DXFOUT_WRITE_TEXT6,IN.Text,IN.Geometrie,IN.Ori,IN.Width,IN.Height
  Plot_Generiert,Skalierungsband_Clipp_Flaeche => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,scaleband_clipp,CONTINUOUS
  Plot_Generiert,Skalierungsband_Border_Linie => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,scaleband_BORDER,CONTINUOUS
  Plot_Generiert,Skalierungsband_Block_Flaeche1 => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,scaleband_BLOCK1,CONTINUOUS
  Plot_Generiert,Skalierungsband_Block_Flaeche2 => DXFOUT_WRITE_POLYLINE2,IN.Geometrie,scaleband_BLOCK2,CONTINUOUS
  Plot_Generiert,Skalierungsband_Text   => DXFOUT_WRITE_TEXT6,IN.Text,IN.Geometrie,IN.Ori,IN.Width,IN.Height
END_MAP

MAP POINT_SYMBIOLOGY
END_MAP

MAP BLOCK_SYMBIOLOGY
  BS_1 => 01131,CONTINUOUS,7,,LFP3ST,0.5
END_MAP

MAP SHAPE_SYMBIOLOGY
END_MAP

MAP TEXT_SYMBIOLOGY
  TS_1 => 01159,CONTINUOUS,7,,STANDARD,1.0,,,
END_MAP

MAP LINE_SYMBIOLOGY
END_MAP

MAP POLYLINE_SYMBIOLOGY
END_MAP

MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
END_MAP

```

```
MAP INOUT
  I1 => IN.TOPIC,IN.TABLE
  I1,FixpunkteKategorie3,LFP3Pos => T_1,TS_1
  I1,FixpunkteKategorie3,LFP3Symbol => B_1,BS_1
  I1,* => OFF

END_MAP

MAP MACRO
  B_1 => DXFOUT_WRITE_BLOCK3,IN.LFP3Symbol_von.Geometrie,0.0
  T_1 => DXFOUT_WRITE_TEXT6,IN.LFP3Pos_von.Nummer,IN.Pos,IN.Ori,IN.HAli,IN.VAli
END_MAP

|INCL \script\plot.mod
|INCL \script\ilin.mod
|INCL \script\dxfout.mod
|INCL \script\runl.prg

! Anwendung plot.mod als Input Source.
! Diese ICS Konfiguration schreibt von ilin.mod
! gelesenen Objekte mit dxfout.mod in eine DXF-Datei
! und schreibt zusätzlich die Plotlayout-Objekte

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP USER_INPUT2
  DIALOG => FILE
  MESSAGE => 'Enter .dxf Output File'
  FILE_FILTER => dxf
  FILE_EXISTS => FALSE
  OPT => output
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\dm01avch24d.ili
  LOG_TABLE => ON
  TRACE => OFF
  STATISTICS => ON
  CALC_SURFACE => ON
END_MAP

MAP DXFOUT_PARAM
  TEMPLATE => \data\geobau2.tem
  CREATE_LAYERS => ON
  STATISTICS => ON
END_MAP

MAP PLOT_PARAM
  TYPE => PLOT
  FORMAT => A4
```

```

FORMATORIENTATION => hoch
ORIGIN              => '675855/245385'
ANGLE               => '45.0'
SCALE               => '1:500'
LAYOUT              => ON
COORDCROSS          => ON
COORDBAND           => ON
SCALEBAND           => ON
END_MAP

MAP PLOT_LAYOUT
A0,quer,*           => \plot\a0q500.itf
A0,hoch,*           => \plot\a0h500.itf
A1,quer,*           => \plot\a1q500.itf
A1,hoch,*           => \plot\a1h500.itf
A2,quer,*           => \plot\a2q500.itf
A2,hoch,*           => \plot\a2h500.itf
A3,quer,*           => \plot\a3q500.itf
A3,hoch,*           => \plot\a3h500.itf
A4,quer,*           => \plot\a4q500.itf
A4,hoch,*           => \plot\a4h500.itf
DEFAULT             => \plot\a4h500.itf
END_MAP

MAP PLOT_COORDCROSS_WIDTH
'1:250'             => 100.0
'1:500'             => 100.0
'1:1000'            => 100.0
DEFAULT             => 100.0
END_MAP

MAP PLOT_SCALEBAND_WIDTH
'1:250'             => 12.0
'1:500'             => 20.0
'1:1000'            => 40.0
END_MAP

MAP PLOT_VALUES
PLOT_PARAM.SCALE   => OPT.massstab
PLOT_DATE           => OPT.datum
'Demogemeinde'     => OPT.gemeinde
END_MAP

MAP POINT_SYBOLOGY
END_MAP

MAP BLOCK_SYBOLOGY
BS_1                => 01131,CONTINUOUS,7,,LFP3ST,0.5
PLOT_BS_1           => coordcross,CONTINUOUS,7,,KOKRZ,IN.Scale
END_MAP

MAP SHAPE_SYBOLOGY
END_MAP

MAP TEXT_SYBOLOGY
TS_1                => 01159,CONTINUOUS,7,,STANDARD,1.0,,,
PLOT_TS_1           => plot,CONTINUOUS,7,,STANDARD,IN.Height,,IN.Slant
PLOT_TS_2           => coordband,CONTINUOUS,7,,STANDARD,IN.Height,,IN.Slant

```

```

PLOT_TS_3 => scaleband,CONTINUOUS,7,,STANDARD,IN.Height,,IN.Slant
END_MAP

MAP LINE_SYMBOLOGY
END_MAP

MAP POLYLINE_SYMBOLOGY
PLOT_PLS_1 => plot,CONTINUOUS,7,,,,
PLOT_PLS_2 => plot_clipp,CONTINUOUS,7,,,,
PLOT_PLS_3 => coordband,CONTINUOUS,7,,,,
PLOT_PLS_4 => scaleband_clipp,CONTINUOUS,7,,,,
PLOT_PLS_5 => scaleband,CONTINUOUS,7,,,,
END_MAP

MAP INPUT_SOURCES
I1 => ILIN,OPT.input
P1 => PLOT
END_MAP

MAP INOUT
I1 => IN.TOPIC,IN.TABLE
I1,FixpunkteKategorie3,LFP3Pos => T_1,TS_1
I1,FixpunkteKategorie3,LFP3Symbol => B_1,BS_1
I1,* => OFF

P1 => IN.TOPIC,IN.TABLE
P1,Plot_Elemente,Border_Flaeche => PLOT_PL_1,PLOT_PLS_1
P1,Plot_Elemente,Clipp_Flaeche => PLOT_PL_1,PLOT_PLS_2
P1,Plot_Elemente,Flaeche => PLOT_PL_1,PLOT_PLS_1
P1,Plot_Elemente,Linie => PLOT_PL_1,PLOT_PLS_1
P1,Plot_Elemente,Text => PLOT_T_1,PLOT_TS_2
P1,Plot_Elemente,Symbol_Linie => PLOT_PL_1,PLOT_PLS_1
P1,Plot_Elemente,Symbol_Flaeche => PLOT_PL_1,PLOT_PLS_1

P1,Plot_Generiert,Koordinatenkreuz => PLOT_B_1,PLOT_BS_1
P1,Plot_Generiert,Koordinatenband_Linie => PLOT_PL_1,PLOT_PLS_3
P1,Plot_Generiert,Koordinatenband_Text => PLOT_T_1,PLOT_TS_2
P1,Plot_Generiert,Skalierungsband_Clipp_Flaeche => PLOT_PL_1,PLOT_PLS_4
P1,Plot_Generiert,Skalierungsband_Border_Linie => PLOT_PL_1,PLOT_PLS_5
P1,Plot_Generiert,Skalierungsband_Block_Flaeche1 => PLOT_PL_1,PLOT_PLS_5
P1,Plot_Generiert,Skalierungsband_Block_Flaeche2 => PLOT_PL_1,PLOT_PLS_5
P1,Plot_Generiert,Skalierungsband_Text => PLOT_T_1,PLOT_TS_3
P1,* => OFF
END_MAP

MAP MACRO
B_1 => DXFOUT_WRITE_BLOCK3,IN.LFP3Symbol_von.Geometrie,0.0
T_1 => DXFOUT_WRITE_TEXT6,IN.LFP3Pos_von.Nummer,IN.Pos,IN.Ori,IN.HAli,IN.Vali

PLOT_B_1 => DXFOUT_WRITE_BLOCK3,IN.Geometrie,0.0
PLOT_T_1 => DXFOUT_WRITE_TEXT6,IN.Text,IN.Geometrie,IN.Ori,IN.HAli,IN.Vali
PLOT_PL_1 => DXFOUT_WRITE_POLYLINE2,IN.Geometrie
END_MAP

|INCL \script\plot.mod
|INCL \script\ilin.mod
|INCL \script\dxfout.mod
|INCL \script\run1.prg

```

10. Modul STAT - Statistiken aus INTERLIS Daten erzeugen

10.1. Allgemeines

Mit dem Modul STAT können statistische Auswertungen aus INTERLIS Daten berechnet werden. Die Statistiken können als Objekte abgefragt oder auch direkt formatiert in die .log Datei geschrieben werden.

STAT wird mit:

```
|INCL \script\stat.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

10.2. Abhängigkeiten von anderen Modulen

Die Inputdaten müssen ilin.mod oder iltopo.mod gelesen werden.

10.3. Parametermap STAT_PARAM

Folgende Parameter können in der Map STAT_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
STATISTIC_DEF	r	Datei mit den Statistikdefinitionen (s.a. unten).
STATISTICS	r	Statistik direkt in die .log Datei schreiben (ON oder OFF). Alternativ dazu können die Statistikobjekte auch mit STAT_READ_OBJECT gelesen werden.

10.4. Objektmodell

Dem Modul STAT werden die INTERLIS-Objekte mit den vordefinierten Prozedur STAT_WRITE_OBJECT0 übergeben (s.a. unten). Nach der Statistikberechnung liefert der STAT Modul folgende IN-Objekte:

Allgemeine Komponenten für jedes IN-Objekt

Komponente	req/opt	Beschreibung
IN.OBJECT(s)	r	Name des INTERLIS-Objekts z.B. Fixpunkte.Fixpunkt.
IN.FUNCTION(s)	r	Name der Statistikfunktion (z.B. COUNT, MIN, MAX, AVG, etc.).
IN.VALUE(n)	r	Wert der Statistikfunktion.
IN.DESCRPTION(s)	r	Beschreibung der Statistikfunktion.

10.5. Exportierte Prozeduren und Methoden

Prozedur STAT_OPEN ! [s arg][[]]

Beschreibung Öffnet den Topologiemodul. Das Argument <arg> hat im Moment keine Bedeutung. STAT_OPEN wird von RUN1 automatisch aufgerufen.

Beispiel	<code>' ' STAT_OPEN</code>
Prozedur	<code>STAT_READ_OBJECT ! [[b status]</code>
Beschreibung	Liest das nächste IN-Objekt gemäss Objektmodell. STAT_READ_OBJECT wird von RUN1 automatisch aufgerufen.
Beispiel	<code>STAT_READ_OBJECT [TRUE]</code>
Prozedur	<code>STAT_CLOSE ! [[]]</code>
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Alle temporären Dateien werden wieder gelöscht. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>STAT_CLOSE</code>
Prozedur	<code>STAT_WRITE_OBJECT0</code>
Beschreibung	Übergibt das aktuelle INTERLIS Objekt dem Statistikmodul.
Beispiel	<code>... => STAT_WRITE_OBJECT0</code>

10.6. Skriptbeispiel

```
! Diese ICS Konfiguration erzeugt Statistiken
! aus den Objekten einer .itf Datei.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .itf Input File'
  FILE_FILTER => itf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP ILIN_PARAM
  INTERLIS_DEF => \models\grunddatensatz.ili
  STATISTICS => OFF
  DEBUG      => OFF
END_MAP

MAP STAT_PARAM
  STATISTIC_DEF => \script\stat\grunddatensatz.sta
  STATISTICS => ON
END_MAP

MAP INPUT_SOURCES
  I1 => ILIN,OPT.input
  I2 => STAT
END_MAP

MAP INOUT
  I1 => STAT_WRITE_OBJECT0
  I2 => OFF
END_MAP
```

```
| INCL \script\util.lib
| INCL \script\stat.mod
| INCL \script\ilin.mod
| INCL \script\run1.prg
```

10.7. Beispiel für Statistikdatei

```
Fixpunkte
  COUNT,016000,ustat_v2_d2_ue2,Anzahl Objekte in Topic Fixpunkte

Fixpunkte.LFP
  COUNT,016001,ustat_v1_d1_ue1,
  MIN,LageGen,016002,ustat_v2_d2_ue2,
  MAX,LageGen,016003,ustat_v2_d2_ue2,
  AVG,LageGen,016004,ustat_v2_d2_ue2,
  STD,LageGen,016005,ustat_v2_d2_ue2,
  MIN,HoeheGen,016006,ustat_v2_d2_ue2,
  MAX,HoeheGen,016007,ustat_v2_d2_ue2,
  AVG,HoeheGen,016008,ustat_v2_d2_ue2,
  STD,HoeheGen,016009,ustat_v2_d2_ue2,
```

11. Modul SURFCUT - Flächenverschnitt

11.1. Allgemeines

Dieses Modul SURFCUT ist durch das allgemeinere Modul INTERSECT abgelöst, welches neben Flächen auch Linien und Punkte verarbeitet.

11.2. Anpassung eines Script vom Modul SURFCUT auf das Modul INTERSECT.

Um eine Konfiguration vom Modul SURFCUT auf das Modul INTERSECT anzupassen, sind folgende Änderungen vorzunehmen.

Anteil	CURFCUT old	INTERSECT new
Parameter Map	MAP SURFCUT_PARAM : END_MAP	MAP INTERSECT_PARAM : END_MAP
Procedures write	MAP INOUT : .. => SURFCUT_WRITE_SURFACE3, ... : END_MAP	MAP INOUT : .. => INTERSECT_WRITE_OBJECT3, ... : END_MAP
Input Sources	MAP INPUT_SOURCES : S => SURFCUT, ... : END_MAP	MAP INPUT_SOURCES : S => INTERSECT, ... : END_MAP

Include	INCL \script\surfcut.mod	INCL \script\intersect.mod
Objekt Attribute	IN.SURFCUT_*	IN.INTERSECT_*

12. Modul TOPO - Topologie berechnen

12.1. Allgemeines

Mit dem Modul TOPO können geschlossene Flächen aus Begrenzungslinien (Boundaries) und Zentroiden berechnet werden. Ausserdem ist es möglich, alle Attribute der angrenzenden Flächen einer Begrenzungslinie oder die Knoten im Flächennetz zu bestimmen. Der Modul wird häufig in Zusammenhang mit Formaten wie AutoCAD DXF oder Microstation DGN benutzt, welche selber nicht den INTERLIS Typ AREA unterstützen. Für INTERLIS 1 und INTERLIS 2 sind die Möglichkeiten des Moduls TOPO bereits in entsprechenden INTERLIS Module eingebaut. Normalerweise muss daher für INTERLIS der Modul TOPO nicht manuell konfiguriert werden.

TOPO wird mit:

```
|INCL \script\topo.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

12.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

12.3. Parametermap TOPO_PARAM

Folgende Parameter können in der Map TOPO_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
RESOLUTION	r	Geometrische Auflösung der Inputgeometrien. Für mm muss z.B. 0.001 eingegeben werden.
OVERLAP	r	Möglicher Overlap für Kreisbögen gemäss INTERLIS Spezifikation (z.B. 0.2).



Die Map TOPO_PARAM ist bereits in topo.mod vordefiniert und muss daher *nicht* in der .cfg Datei definiert werden. Die TOPO_MAP muss daher in einem PRE_SOURCE_* Trigger oder mit NOOP gesetzt werden.

12.4. Objektmodell

Dem Modul TOPO werden Objekte mit den vordefinierten Prozeduren TOPO_WRITE_CENTROID2 bzw. TOPO_WRITE_BOUNDARY1 übergeben (s.a. unten). Nach der Topologieberechnung liefert der TOPO Modul folgende IN-Objekte:

Allgemeine Komponenten für jedes IN-Objekt

Komponente	req/opt	Beschreibung
IN.TTYPE(s)	r	Typ des gelieferten Objekts (s.a. unten).

Zusätzliche Komponenten für IN.TTYPE = 'AREA'

Komponente	req/opt	Beschreibung
IN.AID(i)	r	Jede berechnete Fläche erhält vom TOPO Modul eine eindeutige Nummer.
IN.GEOM(a)	r	Berechnet Fläche.
IN.VALUE(s)	r	Attributwert der Fläche (s.a. TOPO_WRITE_CENTROID2).

Zusätzliche Komponenten für IN.TTYPE = 'BOUNDARY'

Komponente	req/opt	Beschreibung
IN.LEFT(m)	r	Map mit allen Attributen der linken Fläche.
IN.RIGHT(m)	r	Map mit allen Attributen der rechten Fläche.
IN.GEOM(s)	r	Geometrie der Begrenzungslinie (= Boundary).

Zusätzliche Komponenten für IN.TTYPE = 'BAD_BOUNDARY'

Komponente	req/opt	Beschreibung
IN.GEOM(s)	r	Geometrie Begrenzungslinie. Es werden nur Geometrien geliefert welche vom TOPO Modul nicht verarbeitet werden konnten.

Zusätzliche Komponenten für IN.TTYPE = 'NODE'

Komponente	req/opt	Beschreibung
IN.GEOM(p)	r	Knotenkoordinate.
IN.DEGREE(i)	r	Knotengrad. Der Knotengrad gibt an, wieviele Boundaries in dem Knoten zusammen stossen.

12.5. Exportierte Prozeduren und Methoden

Prozedur TOPO_OPEN ! [s type][[]]

Beschreibung Öffnet den Topologiemodul zwecks Berechnung des Topologietyps <type>. <type> kann folgende Werte annehmen:

AREA

Berechnet aus Begrenzungslinien und Zentroidpunkten geschlossene Flächen.

BOUNDARY

Berechnet für jede Begrenzungslinie die Fläche links bzw. rechts der Begrenzungslinie.

BAD_BOUNDARY

Liefert alle fehlerhaften Begrenzungslinien (z.B. Begrenzungslinien mit Lücken).

NODE

Liefert alle Knoten mit Koordinate und Knotengrad.

TOPO_OPEN wird von RUN1 automatisch aufgerufen.

Beispiel 'AREA' TOPO_OPEN

Prozedur	TOPO_READ_OBJECT ! [[b status]
Beschreibung	Liest das nächste IN-Objekt gemäss Objektmodell. TOPO_READ_OBJECT wird von RUN1 automatisch aufgerufen.
Beispiel	<code>TOPO_READ_OBJECT [TRUE]</code>
Prozedur	TOPO_CLOSE ! [[]]
Beschreibung	Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Alle temporären Dateien werden wieder gelöscht. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel	<code>TOPO_CLOSE</code>
Prozedur	TOPO_WRITE_BOUNDARY1 ! l boundary
Beschreibung	Schreibt die Begrenzungslinie <boundary> in den Topologiebuffer.
Beispiel	<code>... => TOPO_WRITE_BOUNDARY1, IN.GEOM</code>
Prozedur	TOPO_WRITE_CENTROID2 ! o value, p centroidpoint
Beschreibung	Schreibt das Zentroid in den Topologiebuffer. Das Zentroid erhält den Attributwert <value> und der Zentroidpunkt ist <centroidpoint>.
Beispiel	<code>... => TOPO_WRITE_CENTROID2, IN.TXT, IN.GEOM</code>

 Es ist auch möglich die Topologie mit der Klasse TOPO ohne die Verwendung von RUN1 zu berechnen (s.a. iG/Script Benutzer- und Referenzhandbuch).

12.6. Skriptbeispiel

```
! Diese ICS Konfiugration berechnet aus DXF Polylines geschlossene
! Flaechen. Die Flaechen werden in der .log Datei angezeigt.
! Fuer die Berechnung wird der TOPO Modul eingesetzt.
```

```
|LICENSE \license\iltools.lic
```

```
MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .dxf Input File'
  FILE_FILTER => dxf
  FILE_EXISTS => TRUE
  OPT => input
END_MAP
```

```
MAP DXFIN_PARAM
  STATISTICS    => ON
  DEBUG        => OFF
END_MAP
```

```
MAP INPUT_SOURCES
  I1 => DXFIN, OPT.input
  I2 => TOPO, AREA
END_MAP
```

```
MAP INOUT
  I1 => IN.LAYER
```

```

I1,01234 => TOPO_WRITE_BOUNDARY1,IN.GEOM
I1,01235 => TOPO_WRITE_CENTROID2,IN.TXT,IN.GEOM
I1,* => OFF
I2 => DISPLAY_OBJECT1,IN
END_MAP

|INCL \script\util.lib
|INCL \script\topo.mod
|INCL \script\dxfin.mod
|INCL \script\run1.prg

```

13. Modul VPRIO - Vektor Elimination nach Prioritäten

13.1. Allgemeines

Mit dem Modul VPRIO können geometrisch übereinanderliegende Vektoren - Linienzüge - über Prioritäten eliminiert werden. Typische Anwendungen sind das Eliminieren von Doppellinien von Flächen die in den INTERLIS 1 Typ AREA übertragen werden müssen oder das Eliminieren von Doppellinien aufgrund der Priorität ermittelt aus einer Linienart.

VPRIO wird mit:

```
|INCL \script\vprio.mod
```

in einer ICS RUN1-Konfiguration verfügbar gemacht.

13.2. Abhängigkeiten von anderen Modulen

Keine Abhängigkeiten vorhanden.

13.3. Parametermap VPRIO_PARAM

Folgende Parameter können in der Map VPRIO_PARAM für den Modul gesetzt werden:

Parameter	req/opt	Beschreibung
RESOLUTION	r	Geometrische Auflösung der Inputgeometrien. Für mm-Auflösung muss z.B. 0.001 eingegeben werden.
TOLERANCE	o	Geometrische Toleranz mit der doppelte Linien eliminiert werden. Wenn der Parameter nicht definiert ist, ist die Tolerance gleich der Resolution.
GEOME- TRY_BREAK_GATTR	o	ON oder OFF, Default = ON. Sollen die zurückzulesenden Geometrien bei unterschiedlichen GATTR-Werten (Geometrie-Attribute) aufgetrennt werden. Das Geometrie-Attribut einer Geometrie kann mit den Methoden ICS.SET_GATTR und ICS.GET_GATTR ggeschrieben und gelesen werden.
GEOME- TRY_BREAK_PRIORI- TY	o	ON oder OFF, Default = OFF. Sollen die zurückzulesenden Geometrien bei unterschiedlichen Prioritäten aufgetrennt werden.
STATISTICS	o	ON oder OFF, Default = OFF. Statistik anzeigen.

13.4. Objektmodell

Dem Modul VPRIO werden Objekte mit der vordefinierten Prozedur `VPRIO_WRITE_LINE2` übergeben (s.a. unten). Nach der Berechnung liefert der Modul VPRIO pro IN-Objekt folgende Systemkomponenten:

Komponente	req/opt	Beschreibung
IN.PRIO(s)	r	Priorität des Objekts.
IN.GEOM(1)	r	Liniengeometrie des Objekts.

13.5. Exportierte Prozeduren und Methoden

Zur Anwendung mit RUN1 stehen folgende Prozeduren zur Verfügung.

Prozedur `VPRIO_WRITE_LINE2 ! g geometry, i priority`
Beschreibung Schreibt die Geometrie mit der Priorität in das Modul. Als Geometrien können Linien- und Flächengeometrien übergeben. Flächengeometrien werden in Liniengeometrien aufgelöst. Eine Geometrie mit der `Priorität=n` eliminiert eine identische Geometrie mit der `Priorität <= n`. Bevor die Elimination mit `VPRIO_OPEN` berechnet wird, müssen mit dieser Prozedur Geometrien in den VPRIO-Buffer geschrieben werden.
Beispiel `... => VPRIO_WRITE_LINE2, IN.GEOM, 3`

Prozedur `VPRIO_OPEN ! [[]]`
Beschreibung Öffnet den Modul zur Berechnung der Elimination der Doppellinien. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel `VPRIO_OPEN`

Prozedur `VPRIO_READ_OBJECT ! [[]b status]`
Beschreibung Liest das nächste IN-Objekt gemäss Objektmodell. Die Prozedur wird von RUN1 automatisch aufgerufen
Beispiel `VPRIO_READ_OBJECT [TRUE]`

Prozedur `VPRIO_CLOSE ! [[]]`
Beschreibung Schliesst den Modul und gibt die durch den Modul belegten Ressourcen wieder frei. Die Prozedur wird von RUN1 automatisch aufgerufen.
Beispiel `VPRIO_CLOSE`

Zur direkten Anwendung ohne RUN1 stehen folgende Methoden zur Verfügung.

Methode `VPRIO.OPEN ! [] []`
Beschreibung Direkte Methode zur Anwendung ohne RUN1. Öffnet den Modul.
Beispiel `VPRIO.OPEN`

Methode `VPRIO.WRITE_LINE ! [g geometry, i priority] []`
Beschreibung Direkte Methode zur Anwendung ohne RUN1. Schreibt die Geometrie mit der Priorität in das Modul. Als Geometrien können Linien- und Flächengeometrien übergeben. Flächengeometrien werden in Liniengeome-

trien aufgelöst. Eine Geometrie mit der Priorität= n eliminiert eine identische Geometrie mit der Priorität $\leq n$. Bevor die Elimination mit `VPRIO.CALCULATE` berechnet wird, müssen mit dieser Methode Geometrien in den `VPRIO-Buffer` geschrieben werden.

Beispiel	<code>VAR.LINE 3 VPRIOWRITE_LINE</code>
Methode	<code>VPRIOWCALCULATE ! [] [b state]</code>
Beschreibung	Direkte Methode zur Anwendung ohne <code>RUN1</code> . Berechnet die Elimination der Doppellinien.
Beispiel	<code>VPRIOWCALCULATE [TRUE]</code>
Prozedur	<code>VPRIOWREAD_LINE ! [] [m map, b status]</code>
Beschreibung	Direkte Methode zur Anwendung ohne <code>RUN1</code> . Liest das nächste Objekt gemäss Objektmodell.
Beispiel	<code>VAR.READ_LINE [m, TRUE]</code>
Methode	<code>VPRIOWCLOSE ! [] []</code>
Beschreibung	Direkte Methode zur Anwendung ohne <code>RUN1</code> . Schliesst den Modul.
Beispiel	<code>VPRIOWCLOSE</code>

Als zusätzliche wichtige Methoden für den Modul sind folgende Methoden aus der Klasse `ICS` hervorzuheben.

Methode	<code>ICSSSET_GATTR [* geometrie, i attr][* geometrie]</code>
Beschreibung	Setzt das Geometrieattribut für die Geometrie <code><geometrie></code> . Vor dem Schreiben einer Geometrie in den Modul kann mit dieser Methode der Geometrie ein Attributwert vergeben werden. Zum Beispiel eine <code>OBJID</code> oder eine Art.
Beispiel	<code>VAR.LINE 1234 ICSSSET_GATTR [line]</code>
Methode	<code>ICSSGET_GATTR [* geometrie][i attr]</code>
Beschreibung	Fragt das Geometrieattribut ab. Nach dem Lesen einer Geometrie aus den Modul kann mit dieser Methode das zusätzliche Attribute der Geometrie gelesen werden. Zum Beispiel eine <code>OBJID</code> oder eine Art.
Beispiel	<code>VAR.LINE ICSSGET_GATTR [1234]</code>

13.6. Skriptbeispiel

```
! Diese ICS Konfiugration liest DXF Polylines und eliminiert
! doppelte Linien. Für die Elimination der doppelten
! Linien wird der Modul VPRIOW eingesetzt.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .dxf Input File'
  FILE_FILTER => dxf
  FILE_EXISTS => TRUE
```

```
OPT => input
END_MAP

MAP DXFIN_PARAM
  STATISTICS => ON
  DEBUG      => OFF
END_MAP

MAP VPRI_PARAM
  RESOLUTION => 0.001
  TOLERANCE  => 0.001
  STATISTICS => ON
END_MAP

MAP INPUT_SOURCES
  I1 => DXFIN,OPT.input
  I2 => VPRI
END_MAP

MAP INOUT
  I1      => IN.LAYER
  I1,01211 => VPRI_WRITE_LINE2,IN.GEOM,1
  I1,01225 => VPRI_WRITE_LINE2,IN.GEOM,2
  I1,*    => OFF
  I2      => DISPLAY_OBJECT1,IN
END_MAP

| INCL \script\dxfin.mod
| INCL \script\vprio.mod
| INCL \script\run1.prg
```

D. iG/Script Bibliotheken

1. Einleitung

In diesem Anhang sind alle iG/Script Bibliotheken und ihre Prozeduren beschrieben.

2. Skriptbibliothek OS.LIB

2.1. Allgemeines

In der Skriptbibliothek OS.LIB sind diverse Hilfprozeduren definiert, welche vom Betriebssystemfunktionen enthalten. Die Bibliothek wird mit:

```
| INCL \script\os.lib
```

in einer ICS Konfiguration verfügbar gemacht.

2.2. Exportierte Prozeduren

Prozedur	<code>COPY_FILE ! [s input][s output]</code>
Beschreibung	Kopiert die Inputdatei <input> nach <output>. Falls die Datei nicht kopiert werden kann, wird der Skript abgebrochen.
Beispiel	<code>'c:\test1.itf' 'c:\test2.itf' COPY_FILE</code>
Prozedur	<code>MAKE_DIR ! [s directory][b state]</code>
Beschreibung	Erzeugt das Dateiverzeichnis <directory>.
Beispiel	<code>IF 'c:\test' MAKE_DIR NOT THEN ERROR 'unable to make directory c:\test' HALT END_IF</code>
Prozedur	<code>REPLACE_BACK_SLASH ! [s string1][s string2]</code>
Beschreibung	Ersetzt alle Backslash Zeichen "\" in <string1> durch Slash.
Beispiel	<code>'c:\test1.itf' REPLACE_BACK_SLASH ! ['c:/test1.itf']</code>
Prozedur	<code>CHANGE_EXTENSION ! [s fname1,s extension][s fname2]</code>
Beschreibung	Ersetzt die Dateierweiterung in <fname1> durch <extension>.
Beispiel	<code>'c:\test1.itf' 'dxf' CHANGE_EXTENSION ! ['c:\test1.dxf']</code>

3. Skriptbibliothek UTIL.LIB

3.1. Allgemeines

In der Skriptbibliothek UTIL sind diverse Hilfprozeduren definiert, welche in einer ICS Konfiguration benutzt werden können (z.B. Prozeduren für die Parameterübernahme in Benutzerprozeduren, Umrechnungsfunktionen für Winkel, Formatierungsprozeduren für Datum und Zeit, Hilfsprozeduren für SQL-Strings, etc.). Die Bibliothek wird mit:

```
|INCL \script\util.lib
```

in einer ICS Konfiguration verfügbar gemacht.

3.2. Exportierte Prozeduren

Prozedur	<code>GRADS_TO_DEGREES ! [r gon][r degree]</code>
Beschreibung	Rechnet Neugrad (400 Gon) in Altgrad (360 Grad) um.
Beispiel	<code>0.0 GRADS_TO_DEGREES ! [90.0]</code>
Prozedur	<code>DEGREES_TO_GRADS ! [r degree][r gon]</code>
Beschreibung	Rechnet Altgrad (360 Grad) in Neugrad (400 Gon) um.
Beispiel	<code>0.0 DEGREES_TO_GRADS ! [100.0]</code>
Prozedur	<code>COPY ! [o obj][o obj]</code>

Beschreibung	Ersetzt das oberste Element des Stacks durch eine exakte Kopie.
Beispiel	<code>'hello, world' COPY ! ['hello, world']</code>
Prozedur	NEXT_OBJID ! [[s objid]
Beschreibung	Erzeugt eine eindeutige OBJID.
Beispiel	<code>NEXT_OBJID ! ['4398']</code>
Prozedur	LAST_OBJID ! [[s objid]
Beschreibung	Gibt die letzte mit NEXT_OBJID erzeugte OBJID zurück.
Beispiel	<code>LAST_OBJID ! ['4398']</code>
Prozedur	DISPLAY_OBJECT1 ! o object
Beschreibung	Gibt <objekt> in der Logdatei aus. DISPLAY_OBJECT1 kann in einer Abbildungsvorschrift der INOUT Map verwendet werden.
Beispiel	<code>MAP INOUT I1 => DISPLAY_OBJECT1,IN END_MAP</code>
Prozedur	SQL_STRING ! [s string][s sql_string]
Beschreibung	Wandelt den <string> in einen SQL-String um, d.h. der String wird zwischen Anführungszeichen gesetzt.
Beispiel	<code>'hello, world' SQL_STRING ! ['hello, world']</code>
Prozedur	GET_PARAM ! [[o object]
Beschreibung	Übernimmt einen Parameter mit beliebigem Typ in einer RUN1 Benutzerprozedur. GET_PARAM soll nur angewendet werden, wenn die Benutzerprozedur auch verschiedene Typen verarbeiten soll. Z.B. ist es sinnvoll, dass eine Anzeigeprozedur beliebige Datentypen anzeigen kann. Sonst sollen aber typspezifische Parameterprozeduren benutzt werden (z.B. GET_SPARAM für die Übernahme eines String Parameters).
Beispiel	<code>PROCEDURE MyDisplayProcedure1 ! message GET_PARAM DISPLAY \$ END_PROCEDURE</code>
Prozedur	GET_SPARAM ! [[s string]
Beschreibung	Übernimmt einen Parameter vom Typ String in einer RUN1 Benutzerprozedur. Falls der Parameter nicht vom Typ String ist, wird die Konfiguration abgebrochen.
Beispiel	<code>PROCEDURE MyStringProcedure1 ! string GET_SPARAM => OUT.Text ! ... ILOUT_WRITE_OBJECT END_PROCEDURE</code>
Prozedur	GET_IPARAM ! [[i integer]
Beschreibung	Übernimmt einen Parameter vom Typ Integer in einer RUN1 Benutzerprozedur. Falls der Parameter nicht vom Typ Integer ist, wird die Konfiguration abgebrochen.

Beispiel	<pre>PROCEDURE MyIntegerProcedure1 ! integer GET_IPARAM => OUT.Count ! ... ILOUT_WRITE_OBJECT END_PROCEDURE</pre>
Prozedur	GET_RPARAM ! [[r real]
Beschreibung	Übernimmt einen Parameter vom Typ Real in einer RUN1 Benutzerprozedur. Falls der Parameter nicht vom Typ Real ist, wird die Konfiguration abgebrochen.
Beispiel	<pre>PROCEDURE MyRealProcedure1 ! real GET_RPARAM => OUT.Area ! ... ILOUT_WRITE_OBJECT END_PROCEDURE</pre>
Prozedur	GET_PPARAM ! [[p point]
Beschreibung	Übernimmt einen Parameter vom Typ Point in einer RUN1 Benutzerprozedur. Falls der Parameter nicht vom Typ Point ist, wird die Konfiguration abgebrochen.
Beispiel	<pre>PROCEDURE MyPointProcedure1 ! point GET_PPARAM => OUT.Geometry ! ... ILOUT_WRITE_OBJECT END_PROCEDURE</pre>
Prozedur	GET_LPARAM ! [[l line]
Beschreibung	Übernimmt einen Parameter vom Typ Line in einer RUN1 Benutzerprozedur. Falls der Parameter nicht vom Typ Line ist, wird die Konfiguration abgebrochen.
Beispiel	<pre>PROCEDURE MyLineProcedure1 ! line GET_LPARAM => OUT.Geometry ! ... ILOUT_WRITE_OBJECT END_PROCEDURE</pre>
Prozedur	GET_APARAM ! [[a area]
Beschreibung	Übernimmt einen Parameter vom Typ Area in einer RUN1 Benutzerprozedur. Falls der Parameter nicht vom Typ Area ist, wird die Konfiguration abgebrochen.
Beispiel	<pre>PROCEDURE MyLineProcedure1 ! area GET_APARAM => OUT.Geometry ! ... ILOUT_WRITE_OBJECT END_PROCEDURE</pre>

4. Skriptbibliothek TRANSFORM.LIB

4.1. Allgemeines

In der Skriptbibliothek TRANSFORM sind diverse Hilfprozeduren definiert, welche in einer ICS Konfiguration benutzt werden können um Transformationen zwischen Koordinatensystemen auszuführen. Die Bibliothek wird mit:

```
| INCL \script\transform.lib
```

in einer ICS Konfiguration verfügbar gemacht. Die Skriptbibliothek ist auch bereits verfügbar, wenn die Skriptbibliothek UTIL inkludiert wird. Die meisten Input- und Output-Module inkludieren die Skriptbibliothek UTIL bereits. Dadurch steht auch die Skriptbibliothek TRANSFORM zur Verfügung.

4.2. Parametermap TRANSFORM_PARAM

Folgende Parameter können in der Map TRANSFORM_PARAM gesetzt werden:

Parameter	req/opt	Beschreibung
SRS_IN	o	Das Input Koordinatensystem.
SRS_OUT	o	Das Output Koordinatensystem.
SRS_FENCE	o	Das Koordinatensystem für eine eventuellen Fence. Der Fence wird von TRANSFORM_PARAM.SRS_FENCE nach TRANSFORM_PARAM.SRS_OUT transformiert.

Die Koordinatensysteme können einer Konfiguration auch mit folgende Optionen übergeben werden:

OPT.srs_in

Entspricht TRANSFORM_PARAM.SRS_IN .

OPT.srs_out

Entspricht TRANSFORM_PARAM.SRS_OUT .

OPT.srs_fence

Entspricht TRANSFORM_PARAM.SRS_FENCE .

Die Koordinatensysteme werden in den Parametern, respektive Optionen wie folgt definiert.

EPSG:<EPSG-Code>

Der Prefix EPSG: ist fix. <EPSG-Code> entspricht dem Code für das Koordinatensystem nach EPSG.

Beispiele:

EPSG:3785 Google Maps

EPSG:4326 WGS84

EPSG:21780 Liechtensteinisches Koordinatensystem FL1903 / LV03

EPSG:21781 Schweizerisches Koordinatensystem CH1903 / LV03

Weitere Koordinatensysteme können bei Bedarf implementiert werden.

4.3. Exportierte Prozeduren

Prozedur	<code>TRANSFORM_GEOMETRY ! [g geometry][g geometry]</code>
Beschreibung	Transformiert eine Geometrie vom Koordinatensystem TRANSFER_PARAM.SRS_IN zum Koordinatensystem TRANSFER_PARAM.SRS_OUT .
Beispiel	<pre>IN.GEOM TRANSFORM_GEOMETRY ! [geometry]</pre>
Prozedur	<code>TRANSFORM_FENCE ! [g geometry][g geometry]</code>
Beschreibung	Transformiert die Geometrie eines eventuellen Fences vom Koordinatensystem TRANSFER_PARAM.SRS_FENCE zum Koordinatensystem TRANSFER_PARAM.SRS_OUT .
Beispiel	<pre>VAR.FENCE TRANSFORM_FENCE ! [geometry]</pre>
Prozedur	<code>TRANSFORM_OBJECT ! [m* map][]</code>
Beschreibung	Transformiert alle Geometrien eines Objektes vom Koordinatensystem TRANSFER_PARAM.SRS_IN zum Koordinatensystem TRANSFER_PARAM.SRS_OUT .
Beispiel	<pre>&IN TRANSFORM_GEOMETRY</pre>
Prozedur	<code>TRANSFORM_OBJECT1 ! o object</code>
Beschreibung	Analog TRANSFORM_OBJECT. Zur Anwendung in der Map INOUT unter run1.
Beispiel	<pre>MAP INOUT : I1 => TRANSFORM_OBJECT1,IN,... : END_MAP</pre>

4.4. Skriptbeispiel

```
! Diese ICS Konfiguration zeigt alle von shpin.mod
! gelesenen Objekte in der .log Datei an.
! Dabei werden die Geometrien der Objekte transformiert.

|LICENSE \license\iltools.lic

MAP USER_INPUT1
  DIALOG => FILE ! FILE | STRING | ODBC
  MESSAGE => 'Enter .shp Input File'
  FILE_FILTER => shp
  FILE_EXISTS => TRUE
  OPT => input
END_MAP

MAP SHPIN_PARAM
  STATISTICS => ON
  DEBUG      => OFF
END_MAP

MAP TRANSFORM_PARAM
  SRS_IN  => 21781
  SRS_OUT => 4326
END_MAP
```

```
MAP INPUT_SOURCES
  I1 => SHPIN,OPT.input
END_MAP

MAP INOUT
  I1 => TRANSFORM_OBJECT1,IN,DISPLAY_OBJECT1,IN,DISPLAY_OBJECT1,SHPIN_REC
END_MAP

|INCL \script\shpin.mod
|INCL \script\run1.prg
```