

# AtlasMap Developer Guide

The AtlasMap Team

Version 1.40.0-SNAPSHOT, 2019-04-26

# AtlasMap Developer Guide

1. Introduction .....	1
2. Quickstart .....	2
2.1. Running AtlasMap Data Mapper UI within Syndesis .....	2
2.2. Running AtlasMap Data Mapper UI standalone .....	2
2.3. Running AtlasMap build .....	2
2.4. Tips for UI developer .....	2
2.4.1. Developing Within Syndesis UI .....	3
2.4.2. Debug unit tests with Chrome DevTools .....	3
2.4.3. Debug Configuration .....	5
3. Internal Design .....	8
3.1. UI .....	8
3.1.1. BOOTSTRAPPING .....	9
3.1.2. MODEL .....	9
3.1.3. SERVICE .....	9
3.2. Design Time Service .....	9
3.2.1. Core Service .....	9
3.2.2. Java Service .....	10
3.2.3. JSON Service .....	10
3.2.4. XML Service .....	10
3.3. Runtime Engine .....	10
3.3.1. AtlasModule and mapping process .....	11
3.3.2. TypeConverter .....	12
3.3.3. FieldAction (Transformation) .....	12
3.3.4. FieldReader .....	12
3.3.5. FieldWriter .....	12
3.3.6. Validation .....	12
3.3.7. Audit .....	12
4. camel-atlasmap component .....	13
5. Reference .....	14
5.1. Design Time Service API .....	14
5.2. Transformation (FieldAction) .....	14
5.3. Java API Reference (Javadoc) .....	23
5.4. Terminology .....	23

# **Chapter 1. Introduction**

This document provides some information for the developers who is willing to contribute to AtlasMap code.

# Chapter 2. Quickstart

## 2.1. Running AtlasMap Data Mapper UI within Syndesis

Data Mapper is primarily designed to be embeded and run in [Syndesis](#) as a Data Mapper step. Simply follow the [Syndesis Developer Handbook](#) to install, and run Syndesis UI. You will find the Data Mapper UI under the integrations panel after selecting or adding an integration with a data mapping step involved in the integration.

## 2.2. Running AtlasMap Data Mapper UI standalone

Here is the shortest path to run standalone AtlasMap.

1. Download AtlasMap standalone jar

```
wget http://central.maven.org/maven2/io/atlasmap/atlasmap-standalone/${VERSION}/atlasmap-standalone-${VERSION}.jar
```

2. Run AtlasMap standalone

```
$ java -jar atlasmap-standalone-${VERSION}.jar
```

Now AtlasMap Data Mapper UI is available at <http://127.0.0.1:8585/>

## 2.3. Running AtlasMap build

Building everything for standalone usage

1. Clone AtlasMap repository

```
$ git clone https://github.com/atlasmap/atlasmap ${ATLASMAP}
```

2. Build AtlasMap runtime

```
$ cd ${ATLASMAP}  
$ ./mvnw clean install
```

## 2.4. Tips for UI developer

## 2.4.1. Developing Within Syndesis UI

Data Mapper UI is referenced by Syndesis as a dependency. When Syndesis UI's dependencies are installed during `yarn install` step, Data Mapper UI will be cloned from the NPM package repository into `SYNDESIS}/app/ui/node_modules/@atlasmap/atlasmap-data-mapper` directory.

You can point your local Syndesis UI's Data Mapper UI reference to your working copy of the Data Mapper by [yarn link](#). You'll do something like this:

```
$ cd ${ATLASMAP}/ui
$ yarn build:lib
$ cd dist/lib
$ yarn link
$ cd ${SYNDESIS}/app/ui
$ yarn link @atlasmap/atlasmap-data-mapper
```

Note that running `yarn install` in the Syndesis UI directory **will remove and redownload the `SYNDESIS}/app/ui/node_modules/@atlasmap/atlasmap-data-mapper` directory**, so you'd want to avoid doing it while you have `yarn link`. Instead you'd change `ATLASMAP}/ui/src` and run `yarn build:lib` to make a change in AtlasMap UI and consume it within Syndesis UI.

## 2.4.2. Debug unit tests with Chrome DevTools

To debug AtlasMap Data Mapper UI unit tests with using Chrome DevTools:

```
$ cd ${ATLASMAP}/ui
$ yarn test:debug
```

Or you can specify target test file alone

```
$ yarn test:debug --main src/app/lib/atlasmap-data-mapper/services/mapping-
serializer.service.spec.ts
```

Then Chromium window opens up. Push the `DEBUG` button. New `DEBUG` tab will open.



Then open Chrome DevTools, open TypeScript source file and put a breakpoint.

Jasmine 2.6.4 finished in 1.374s

58 specs, 0 failures raise exceptions

```

  Jasmine 2.6.4
  finished in 1.374s
  58 specs, 0 failures
  raise exceptions

  AppComponent
  should ...
  AtlassapNavbarComponent
  should ...
  ExampleAppModule
  should ...
  DataMapperUtil
  should ...
  ClassNameComponent
  should ...
  CollapsibleHeaderComponent
  should ...
  ConstantFieldEditComponent
  should ...
  DataMapperAppComponent
  should ...
  DataMapperErrorComponent
  should ...
  DataMapperAppExampleHostComponent
  should ...
  DocumentDefinitionComponent
  should ...
  DocumentFieldDetailComponent
  should ...
  FieldEditComponent
  should ...
  LineMachineComponent
  should ...
  LookUpEditComponent
  should ...
  MappingDetailComponent
  should ...
  MappingFieldActionComponent
  should ...
  MappingFieldDetailComponent
  should ...
  MappingListComponent
  should ...
  MappingSelectionComponent
  should ...
  TransitionSelectionComponent
  should ...
  ModalWindowComponent
  should be initialized with EmptyModalBodyComponent
  should load ConstantFieldEditComponent
  should invoke OK button handler
  should invoke cancel button handler

```

Developer Tools - http://localhost:9876/debug.html

Paused on breakpoint

Filesystem Network Elements Console Sources Network Performance Memory Application Security Audits

modal-window.components.ts:1

```

1 Serving from the file system? Add your files into the workspace. more never show
2 Watch
3 Call Stack
4 Not paused
5 Scope
6 Not paused
7 Breakpoints
8 modal-window.component.ts:84
9 this.loadComponent();
10 XHR/Fetch Breakpoints
11 DOM Breakpoints
12 Global Listeners
13 Event Listener Breakpoints
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
813
814
815
816
817
818
819
819
820
821
822
823
823
824
825
826
827
828
828
829
829
830
831
832
833
833
834
835
836
837
837
838
839
839
840
841
842
842
843
844
844
845
846
846
847
847
848
848
849
849
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
```

2. **baseXMLInspectionServiceUrl** - URL for the XML Inspection Service provided by the AtlasMap Services.
3. **baseJSONInspectionServiceUrl** - URL for the JSON Inspection Service provided by the AtlasMap Services.
4. **baseMappingServiceUrl** - URL for the Mapping Service provided by the AtlasMap Services.

## Mock Source/Target Document Configuration

These flags control the UI automatically adding the specified mock documents to the system when the UI initializes.

1. **addMockJavaSingleSource** - Add single Java source document.
2. **addMockJavaSources** - Add multiple Java source documents.
3. **addMockXMLInstanceSources** - Add multiple XML instance-based source documents.
4. **addMockXMLSchemaSources** - Add multiple XML schema-based source documents.
5. **addMockJSONSources** - Add multiple JSON source documents.
6. **addMockJavaTarget** - Add a Java target document.
7. **addMockXMLInstanceTarget** - Add a XML instance target document.
8. **addMockXMLSchemaTarget** - Add a XML schema target document.
9. **addMockJSONTarget** - Add a JSON target document.

The code that initializes these mock documents is in the [InitializationService](#). That service calls various static methods in the [DocumentManagementService](#) to create example XML instance, XML schema, and JSON documents. Mock Java documents referenced are from the AtlasMap Services' [Atlas Java Test Model Maven Module](#).

## Additional Debug Configuration

1. **discardNonMockSources** - Automatically discard all user-specified (or Syndesis UI-specified) source/target documents before initializing. This is helpful if you're trying to test with mock documents alone.
2. **addMockJSONMappings** - This flag bootstraps the UI's mappings from the provided JSON mapping definition. Useful for repeatedly debugging a particular scenario.
3. **debugClassPathServiceCalls** - Log details about JSON request/responses to/from the class path resolution service.
4. **debugDocumentServiceCalls** - Log details about JSON request/responses to/from the Java/XML/JSON inspection services.
5. **debugMappingServiceCalls** - Log details about JSON request/responses to/from the mapping service.
6. **debugValidationServiceCalls** - Log details about JSON request/responses to/from the mapping validation service.
7. **debugFieldActionServiceCalls** - Log details about JSON request/responses to/from the mapping field action configuration service.

8. **debugDocumentParsing** - Log details about parsing JSON responses from the inspection services.

Data Mapper Debug Configuration within the Syndesis UI is defined within your \${SYNDESIS}/ui/src/config.json file's data mapper section:

```
{  
  "apiEndpoint": "https://syndesis-staging.b6ff.rh-idev.openshiftapps.com/api/v1",  
  "title": "Syndesis",  
  "datamapper": {  
    "baseJavaInspectionServiceUrl": "http://localhost:8585/v2/atlas/java/",  
    "baseXMLInspectionServiceUrl": "http://localhost:8585/v2/atlas/xml/",  
    "baseJSONInspectionServiceUrl": "http://localhost:8585/v2/atlas/json/",  
    "baseMappingServiceUrl": "http://localhost:8585/v2/atlas/",  
    "discardNonMockSources": true,  
    "addMockJSONMappings": false,  
    "addMockJavaSingleSource": true,  
    "addMockJavaSources": false,  
    "addMockXMLInstanceSources": true,  
    "addMockXMLSchemaSources": true,  
    "addMockJSONSources": true,  
    "addMockJavaTarget": false,  
    "addMockXMLInstanceTarget": false,  
    "addMockXMLSchemaTarget": false,  
    "addMockJSONTarget": true,  
    "debugDocumentServiceCalls": true,  
    "debugMappingServiceCalls": true,  
    "debugClassPathServiceCalls": false,  
    "debugValidationServiceCalls": false,  
    "debugFieldActionServiceCalls": false,  
    "debugDocumentParsing": false  
  },  
  "oauth": {  
    "clientId": "syndesis-ui",  
    "scopes": ["openid"],  
    "oidc": true,  
    "hybrid": true,  
    "issuer": "https://syndesis-staging.b6ff.rh-idev.openshiftapps.com/auth/realms/syndesis",  
    "auto-link-github": true  
  }  
}
```

If you're running the Data Mapper UI locally outside of the Syndesis UI, the debug configuration is specified within the [DataMapperAppExampleHostComponent](#).

# Chapter 3. Internal Design

AtlasMap consists of following 3 parts:

1. [Data Mapper UI](#)
2. [Design Time Service](#)
3. [Runtime Engine](#)

[Data Mapper UI](#) is an Angular based web browser application. [Design Time Service](#) is a set of REST API which provide background services to be used by the Data Mapper UI behind the scene. Data Mapper UI eventually produces data mapping definition file in XML or JSON format. Then [Runtime Engine](#) consumes that mapping definition as well as actual data payload and perform mapping.

When data formats are provided into Data Mapper UI, it requests an inspection to Design Time Service and receives a unified metadata, called Document. For example, if the data format is provided as a JSON schema, Data Mapper UI makes a JSON schema inspection request and receive a Document object. While JSON schema is a JSON specific schema to represent message payload, Document object is an AtlasMap internal, but data format agnostic metadata so that the Data Mapper UI can consume and provide an unified mapping experience.

AtlasMap Document object defines a tree of fields. Defining a set of field-to-field mapping is all about the mappings in AtlasMap.

Another thing to know for AtlasMap internal is the modules located [here](#) in repository. Module in AtlasMap is a facility to plug-in an individual data format support like Java, JSON and XML. Each module implements roughly following 2 parts:

- Inspection Design Time Service to convert the format specific metadata into AtlasMap Document object
- Runtime SPI to achieve actual mappings
  - [AtlasModule](#): Several methods to be invoked during processing mappings. We'll look into deeper in [AtlasModule](#) section.
  - [AtlasFieldReader](#): Read a field value from source payload
  - [AtlasFieldWriter](#): Write a field value into target payload



There is also an overview document for the Data Mapper step in Syndesis, which might help if you look into AtlasMap within Syndesis context. <https://github.com/syndesisio/syndesis/blob/master/app/server/docs/design/datamapper.md>

## 3.1. UI

Data Mapper UI is a web based user interface to define a data mapping, built with [Angular](#).

- <https://github.com/atlasmap/atlasmap/blob/master/ui/>

### 3.1.1. BOOTSTRAPPING

Bootstrapping the Data Mapper UI requires a bit of configuration. An example bootstrapping component is provided within the project:

<https://github.com/atlasmap/atlasmap/blob/master/ui/src/app/lib/atlasmap-data-mapper/components/data-mapper-example-host.component.ts>

### 3.1.2. MODEL

All application data and configuration is stored in a centralized ConfigModel object.

The ConfigModel contains:

- initialization data such as service URLs and source/target document information
- references to our angular2 services that manage retrieving and saving our documents and mapping data
- document / mapping model objects

There are two document models contained within the ConfigModel object, both of type DocumentDefinition. A DocumentDefinition contains information about a source or target document such as the document's name, and fields for that document. Fields are represented by our Field model.

A single MappingDefinition model in the ConfigModel object stores information about field mappings and related lookup tables. Individual mappings are represented in instances of MappingModel, and lookup tables are represented by the LookupTable model.

### 3.1.3. SERVICE

When the Data Mapper UI Bootstraps, a series of service calls are made to the mapping service ([MappingManagementService](#)) and document service ([DocumentManagementService](#)).

The document service is used to fetch our source/target document information (name of doc, fields). After these are parsed from the service, they are stored in the ConfigModel's inputDoc and outputDoc DocumentDefinition models.

The mapping service is used to fetch our mappings for the fields mapped from the source to the target document. These mappings (and related lookup tables) are parsed by the management service and stored in the ConfigModel's mappings MappingDefinition model.

## 3.2. Design Time Service

Design Time Service is a set of REST API which provide background services to be used by the Data Mapper UI behind the scene. There are 4 types of Design Time Service:

### 3.2.1. Core Service

- Code Location: <https://github.com/atlasmap/atlasmap/blob/master/lib/service>

- API Reference: [Core Service](#)

Core Service provides basic operations which is not specific to the individual data formats, Create/Get/Update/Remove mapping definition stored in Design Time Service local storage, validate mapping, retrieve metadata for available field actions and etc.

### 3.2.2. Java Service

- Code Location: <https://github.com/atlasmap/atlasmap/blob/master/lib/modules/java/service>
- API Reference: [Java Service](#)

Java Service provides Java inspection service which generate an AtlasMap Document object from Java class name.

### 3.2.3. JSON Service

- Code Location: <https://github.com/atlasmap/atlasmap/blob/master/lib/modules/json/service>
- API Reference: [JSON Service](#)

JSON Service provides JSON inspection service which generate an AtlasMap Document object from JSON instance or JSON schema.

### 3.2.4. XML Service

- Code Location: <https://github.com/atlasmap/atlasmap/blob/master/lib/modules/xml/service>
- API Reference: [XML Service](#)

XML Service provides XML inspection service which generate an AtlasMap Document object from XML instance or XML schema.

## 3.3. Runtime Engine

- Code Location: <https://github.com/atlasmap/atlasmap/blob/master/lib/>

AtlasMap runtime engine consumes mapping definition file created via [Data Mapper UI](#) as well as actual data payload and perform mapping.

Here is a shortest code to process a mapping using AtlasMap runtime engine:

```
AtlasContextFactory factory = atlasContextFactory = DefaultAtlasContextFactory
.getInstance(); ①
AtlasContext context = factory.createContext(new File("./my-mapping.xml")); ②
AtlasSession session = context.createSession(); ③
session.setSourceDocument("myJsonSourceDoc", "{...}"); ④
context.process(session); ⑤
Object targetDoc = session.getTargetDocument("myXmlTargetDoc"); ⑥
```

① `AtlasContextFactory` is a singleton instance on JVM, which holds global configuration for the

AtlasMap.

- ② `AtlasContextFactory#createContext(File)` creates `AtlasContext` which represents an `AtlasMap` mapping context for each mapping definitions by feeding a mapping definition file.
- ③ `AtlasSession` represents a mapping processing session. `AtlasSession` should be created for each execution and should NOT be shared among multiple threads.
- ④ Put a source Document with a corresponding Document ID. Make sure Document ID matches with what is specified in the mapping definition.
- ⑤ Process a mapping by `AtlasContext#process(AtlasSession)`. This invocation also triggers `mapping validation` prior to actually perform a mapping.
- ⑥ Finally take the transformed document out from `AtlasSession` by specifying target Document ID.

### 3.3.1. `AtlasModule` and mapping process

`AtlasModule` is a SPI to be implemented by each modules like `Java`, `JSON` and `XML`. The methods defined in `AtlasModule` are invoked from `AtlasContext` while `AtlasContext#process(AtlasSession)` is in progress.

`AtlasContext#process(AtlasSession)` goes on in following order:

1. `AtlasContext#processValidation(AtlasSession)`
  - a. `AtlasValidationService.validateMapping(AtlasMapping)` validates mapping definition
  - b. `AtlasModule#processPreValidation(AtlasSession)` for each modules participated in the mapping, validates data format specific things
2. `AtlasModule#processPreSourceExecution(AtlasSession)` for each source modules
3. `AtlasModule#processPreTargetExecution(AtlasSession)` for each target modules
4. for each mapping entries:
  - a. `AtlasModule#processSourceFieldMapping(AtlasSession)` for each source fields
    - i. Read source field values from source payload with using `FieldReader`
    - ii. Apply `FieldAction` if it's specified for the source field
  - b. `AtlasModule#processTargetFieldMapping(AtlasSession)` for each target fields
    - i. Convert source field values into target field type with using `TypeConverter` if needed
    - ii. Copy source field values into target fields
    - iii. Apply `FieldAction` if it's specified for the target field
    - iv. Write target field values into target payload with using `FieldWriter`
5. `AtlasModule#processPostValidation(AtlasSession)` for each modules
6. `AtlasModule#processPostSourceExecution(AtlasSession)` for each source modules
7. `AtlasModule#processPostTargetExecution(AtlasSession)` for each target modules

### 3.3.2. TypeConverter

TypeConverter converts one field value to the expected field type. This is automatically invoked during mapping when the actual value is not in expected type. AtlasMap runtime provides OOTB converters for the AtlasMap primitive types [here](#).

### 3.3.3. FieldAction (Transformation)

FieldAction is a function you can apply on a field value as a part of mapping. AtlasMap provides a variety of FieldActions you can apply in the middle of processing mappings [here](#). Also There is a [Reference](#) for all available FieldAction.

### 3.3.4. FieldReader

Each module implements its own [FieldReader](#) to read a field value from document specific payload.

### 3.3.5. FieldWriter

Each module implements its own [FieldWriter](#) to write a field value into document specific payload.

### 3.3.6. Validation

`AtlasContext#processValidation(AtlasSession)` validates a mapping definition associated with this context. After it's completed, you can retrieve a collection of [Validation](#) object which represents a validation log.

`processValidation(AtlasSession)` is also invoked as a part of `AtlasContext#process(AtlasSession)` prior to actually perform a mapping. In this case, validation results are converted to [Audit](#).

### 3.3.7. Audit

[Audit](#) represents an audit log which is emitted from runtime engine during processing a mapping. After `AtlasContext#process(AtlasSession)` is completed, you can retrieve a collection of [Audit](#) object by invoking `AtlasSession.getAudits()`.

# Chapter 4. camel-atlasmap component

camel-atlasmap is an [Apache Camel](#) Component for AtlasMap. This component executes AtlasMap mapping as a part of Camel route processing.

Example usage:

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from uri="direct:start" />
        <to uri="atlas:atlasmapping.xml" />
        <to uri="mock:result" />
    </route>
</camelContext>
```

# Chapter 5. Reference

## 5.1. Design Time Service API

## 5.2. Transformation (FieldAction)

Transformation is a function you can apply on a field value as a part of mapping. There are a variety of transformations you can apply in the middle of processing mappings. This is called FieldAction internally.



TODO: Generate this list automatically from annotation - <https://github.com/atlasmap/atlasmap/issues/173>

Field Action	Input Type	Output Type	Parameter(s) (*=required)	Description
AbsoluteValue	Number	Number	n/a	Return the absolute value of a number.
Add	Collection/Array/Map	Number	n/a	Add the numbers in a collection, array, or map's values.
AddDays	Date	Date	days	Add days to a date. The default days is 0.
AddSeconds	Date	Date	seconds	Add seconds to a date. The default seconds is 0.
Append	String	String	string	Append a string to the end of a string. The default is to append nothing.
Average	Collection/Array/Map	Number	n/a	Return the average of the numbers in a collection, array, or map's values.

Field Action	Input Type	Output Type	Parameter(s) (*=required)	Description
Camelize	String	String	n/a	Convert a phrase to a camelized string by: Removing whitespace Making the first word lowercase Capitalizing the first letter of each subsequent word
Capitalize	String	String	n/a	Capitalize the first character of a string.
Ceiling	Number	Number	n/a	Return the whole number ceiling of a number.
Concatenate	Collection/Array/Map	String	delimiter	Concatenate a collection, array, or map's values, separating each entry by the delimiter if supplied.
Contains	Any	Boolean	value	Return true if a field contains the supplied value.
ConvertAreaUnit	Number	Number	fromUnit *toUnit *	Convert a number representing an area to another unit. The fromUnit and toUnit parameters can be one of the following: Square FootSquare MeterSquare Mile

Field Action	Input Type	Output Type	Parameter(s) (*=required)	Description
ConvertDistanceUnit	Number	Number	fromUnit *toUnit *	Convert a number representing a distance to another unit. The fromUnit and toUnit parameters can be one of the following: FootInchMeter MileYard
ConvertMassUnit	Number	Number	fromUnit *toUnit *	Convert a number representing a mass to another unit. The fromUnit and toUnit parameters can be one of the following: KilogramPound
ConvertVolumeUnit	Number	Number	fromUnit *toUnit *	Convert a number representing a volume to another unit. The fromUnit and toUnit parameters can be one of the following: Cubic footCubic meterGallonLiter
CurrentDate	n/a	Date	n/a	Return the current date.
CurrentDateTime	n/a	Date	n/a	Return the current date/time.
CurrentTime	n/a	Date	n/a	Return the current time.

Field Action	Input Type	Output Type	Parameter(s) (*=required)	Description
DayOfWeek	Date	Number	n/a	Return the day of the week for a date, from 1 (Monday) to 7 (Sunday).
DayOfYear	Date	Number	n/a	Return the day of the year for a date, from 1 to 365 (or 366 in a leap year).
Divide	Collection/Array/Map	Number	n/a	Divide each entry in a collection, array, or map's values by its subsequent entry (A "normal" division would only involve 2 entries).
EndsWith	String	Boolean	string	Return true if a string ends with the supplied string (including case).
Equals	Any	Boolean	value	Return true if a field is equal to the supplied value (including case).
FileExtension	String	String	n/a	Retrieve the extension, without the dot ('.'), of a string representing a file name.
Floor	Number	Number	n/a	Return the whole number floor of a number.

Field Action	Input Type	Output Type	Parameter(s) (*=required)	Description
Format	Any	String	template *	Return a string that is the result of substituting a field's value within a template containing placeholders like %s, %d, etc., similar to mechanisms available in programming languages like Java and C.
GenerateUUID	n/a	String	n/a	Create a string representing a random UUID.
IndexOf	String	Number	string	Return the first index, starting at 0, of the supplied string within a string, or -1 if not found.
IsNull	Any	Boolean	n/a	Return true if a field is null.
LastIndexOf	String	Number	string	Return the last index, starting at 0, of the supplied string within a string, or -1 if not found.
Length	Any	Number	n/a	Return the length of the field, or -1 if null. For collections, arrays, and maps, this means the number of entries.

Field Action	Input Type	Output Type	Parameter(s) (*=required)	Description
Lowercase	String	String	n/a	Convert a string to lowercase.
Maximum	Collection/Array/Map	Number	n/a	Return the maximum number from the numbers in a collection, array, or map's values.
Minimum	Collection/Array/Map	Number	n/a	Return the minimum number from the numbers in a collection, array, or map's values.
Multiply	Collection/Array/Map	Number	n/a	Multiply the numbers in a collection, array, or map's values.
Normalize	String	String	n/a	Replace consecutive whitespace characters with a single space and trim leading and trailing whitespace from a string.
PadStringLeft	String	String	padCharacter *padCount *	Insert the supplied character to the beginning of a string the supplied count times.
PadStringRight	String	String	padCharacter *padCount *	Insert the supplied character to the end of a string the supplied count times.

Field Action	Input Type	Output Type	Parameter(s) (*=required)	Description
Prepend	String	String	string	Prepend a string to the beginning of a string. The default is to prepend nothing.
ReplaceAll	String	String	match *newString	Replace all occurrences of the supplied matching string in a string with the supplied newString. The default newString is an empty string.
ReplaceFirst	String	String	match *newString	Replace this first occurrence of the supplied matching string in a string with the supplied newString. The default newString is an empty string.
Round	Number	Number	n/a	Return the rounded whole number of a number.
SeparateByDash	String	String	n/a	Replace all occurrences of whitespace, colons (:), underscores (_), plus (+), or equals (=) with a dash (-) in a string.

<b>Field Action</b>	<b>Input Type</b>	<b>Output Type</b>	<b>Parameter(s) (*=required)</b>	<b>Description</b>
SeparateByUnder score	String	String	n/a	Replace all occurrences of whitespace, colon (:), dash (-), plus (+), or equals (=) with an underscores (_) in a string.
StartsWith	String	Boolean	string	Return true if a string starts with the supplied string (including case).
Substring	String	String	startIndex *endIndex	Retrieve the segment of a string from the supplied inclusive startIndex to the supplied exclusive endIndex. Both indexes start at zero. The default endIndex is the length of the string.

Field Action	Input Type	Output Type	Parameter(s) (*=required)	Description
SubstringAfter	String	String	startIndex *endIndexmatch *	Retrieve the segment of a string after the supplied match string from the supplied inclusive startIndex to the supplied exclusive endIndex. Both indexes start at zero. The default endIndex is the length of the string after the supplied match string.
SubstringBefore	String	String	startIndex *endIndexmatch *	Retrieve the segment of a string before the supplied match string from the supplied inclusive startIndex to the supplied exclusive endIndex. Both indexes start at zero. The default endIndex is the length of the string before the supplied match string.

Field Action	Input Type	Output Type	Parameter(s) (*=required)	Description
Subtract	Collection/Array/Map	Number	n/a	Subtract each entry in a collection, array, or map's values from its previous entry (A "normal" subtraction would only involve 2 entries).
Trim	String	String	n/a	Trim leading and trailing whitespace from a string.
TrimLeft	String	String	n/a	Trim leading whitespace from a string.
TrimRight	String	String	n/a	Trim trailing whitespace from a string.
Uppercase	String	String	n/a	Convert a string to uppercase.

## 5.3. Java API Reference (Javadoc)

## 5.4. Terminology

AtlasMap terminology

Term	Definition	Example
AtlasMapping	The top-level mapping file containing the mapping instructions to execute at runtime	atlasmapping.xml, atlasmapping.json
Audit	An Audit is informational tracing data captured runtime execution in order to provide user feedback	Useful during testing cycles, or when user does not have direct access to the runtime logs
Collection	A data type used in conversion that repeats 0 or more times	Used to process Arrays and Lists
Data Source	The definition of an input or output data format within an AtlasMapping file	atlas:java, atlas:json, atlas:xml, etc

Term	Definition	Example
Field	The base type used to describe a data value used in a Mapping	JavaField, JsonField, XmlField, etc
Field Action	A function to perform on a given value of an input or output field	CurrentDate, Trim, SubString, etc
Field Type	Normalized convention to describe the type of data expected within the value of a field	String, Integer, Long, Number, Date Time, etc
Lookup Table	A cross reference table used at runtime to define a data conversion that varies based on the value of the input field	Used for mapping enumerations
Mapping	A grouping of input Field(s), output Field(s) and optionally Field Action(s) that defines the conversion to be performed at runtime within an AtlasMapping file	JavaField → JsonField, XmlField → JsonField + Uppercase, etc
Mapping Type	The specific type of conversion that should be performed at runtime	Map, Combine, Separate, Collection or Lookup
Validation	A Validation is a syntax check of a provide mapping file to ensure execution may proceed	Useful during testing cycles, and at runtime to avoid complicated and misleading exception stack traces

## AtlasMap Data Format terminology

Term	Definition	Example
Boxed Primitive	A type of primitive that have a 'null' value	Integer, Long, Boolean, etc
(Unboxed) Primitive	A type of primitive that cannot be 'null' and generally has a default initialized value	int, long, boolean, etc
Rooted	JSON documents that have a parent node identifying to contents of the JSON document	{ "Contact": { "firstName": "Yu", "lastName": "Darvish" } }
Unrooted	JSON documents that do not have a parent node identifying the contents of the JSON document	{ "firstName": "Yu", "lastName": "Darvish" }
Qualified	Xml elements and/or attributes that contain the namespace prefix	<ns1>Contact ns1:firstName="Yu" ns1:lastName="Darvish" />
Unqualified	Xml elements and/or attributes that do not contain the namespace prefix	<Contact firstName="Yu" lastName="Darvish" />