

AN INTRO TO GETTING DYNAMO TO TALK WITH RAM STRUCTURE USING THE API

Marcello Sgambelluri S.E.

Director of Advanced Technology

Twitter: @marcellosgamb

Blog/Podcast/Comic: simplycomplex.org





About the speaker

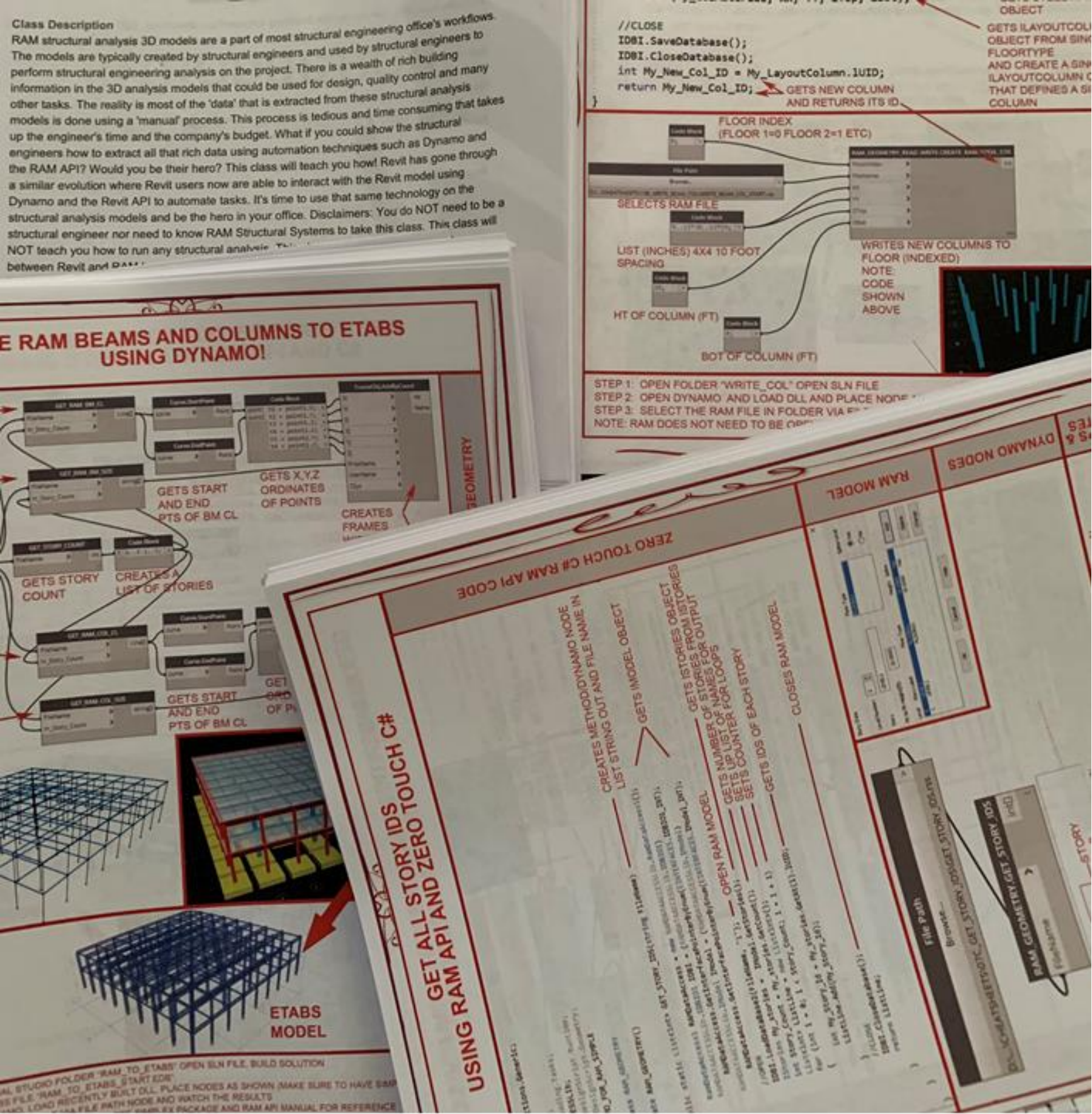
Marcello Sgambelluri S.E.

Director of Advanced Technology

Marcello has worked on many BIM projects over the last 20 years and continually speaks at Autodesk University and has received the 1st place speaker award a record 6 times between 2012 thru 2018.

Marcello Sgambelluri is a licensed civil and structural engineer and helps the AEC Community thru Training, Conferences, Blog, Podcast, Youtube, and Comics

<https://www.simplycomplex.org/>



Get Handouts
And DataSet
from the
AU app or
HERE



HANDOUTS

Short Version 36 Pgs

Long Version 200 Pgs

<https://a360.co/2MQuFPO>

	_000_ZTSTARTER	7/13/2019 9:13 PM	File folder
	_000RAM_API_ORGANIZATION_CHART	7/5/2019 10:10 PM	File folder
	_001_GET_STORY_COUNT	7/14/2019 6:50 PM	File folder
	_002_GET_BEAM_COLUMNS_SIZES	7/14/2019 6:51 PM	File folder
	_003_GET_BEAM_COL_GEOMETRY_N_QA_QC	7/5/2019 10:11 PM	File folder
	_004_GET_GRIDS_AT_COL_TO_EXCEL	7/9/2019 6:36 PM	File folder
	_005_BEAM_DESIGN_OUTPUT	7/8/2019 6:34 PM	File folder
	_006_WRITE_TO_RAM_FLOORTYPE	7/14/2019 7:03 PM	File folder
	_007_RAM_TO_RAM_VERTBRACE	7/9/2019 6:38 PM	File folder
	_008_RAM_TO_ETABS_COL_BM	7/5/2019 10:13 PM	File folder
	_009_REVIT_TO_RAM_GRID	7/5/2019 10:13 PM	File folder
	_ETABS_EXAMPLES	7/5/2019 10:17 PM	File folder
	001_ZERO_TOUCH_BASICS_STARTING_A_VISUAL_STUDIO_PROJECT	7/5/2019 10:04 PM	File folder
	002_ZERO_TOUCH_BASICS_LOADING_IN_REFERENCES	7/5/2019 10:09 PM	File folder
	003_ZERO_TOUCH_BASICS_ANATOMY_OF_ZT_NODE_NO_INPUT	7/5/2019 10:09 PM	File folder
	004_ZERO_TOUCH_BASICS_ANATOMY_OF_ZT_NODE_INPUT	7/5/2019 10:09 PM	File folder
	005_ZERO_TOUCH_BASICS_GET_AGE	7/5/2019 10:10 PM	File folder
	007_RAM_FILE_ERROR_FIX	7/5/2019 10:10 PM	File folder
	009_GET_STORY_NAMES	7/5/2019 10:10 PM	File folder
	010_GET_STORY_IDS	7/8/2019 5:41 PM	File folder
	011_GET_NUM_COLUMNS	7/5/2019 10:11 PM	File folder
	013_EXTRACT_LINES_BEAM_COLUMNS_CODE	7/5/2019 10:11 PM	File folder
	015_GET_BEAM_COLUMNS_IDS	7/8/2019 5:43 PM	File folder
	017_GET_BEAM_COL_IS_GRAV_LAT	7/5/2019 10:12 PM	File folder
	018_WRITE_COL	7/5/2019 10:12 PM	File folder
	019_CHANGE_BEAM_LOC	7/5/2019 10:12 PM	File folder
	020_WRITE_BEAM	7/5/2019 10:12 PM	File folder
	021_WRITE_HOR_BRACE	7/5/2019 10:12 PM	File folder
	022_WRITE_VERT_BRACE	7/5/2019 10:12 PM	File folder
	024_ETABS_TO_RAM_COL_BM	7/5/2019 10:13 PM	File folder
	025_GET_GRID	7/5/2019 10:13 PM	File folder
	026_WRITE_GRID	7/5/2019 10:13 PM	File folder
	028_COLUMN_FORCES	7/8/2019 5:41 PM	File folder

DATASET

EVERY EXAMPLE+ SOURCE C# CODE

Session Description

RAM structural analysis 3D models are a part of most structural engineering office's workflows. There is a wealth of rich building information in the 3D analysis models that could be used for design, quality control and many other tasks. The reality is most of the 'data' that is extracted from these structural analysis models is done using a 'manual' process. This process is tedious and time consuming that takes up the engineer's time and the company's budget. What if you could show the structural engineers how to extract all that rich data using automation techniques such as Dynamo and the RAM API? Would you be their hero?

This class will NOT discuss any links between Revit and RAM because this class WILL focus on how to 'extract' EXISTING data from EXISTING RAM models.

Learning Objectives

At the end of this session, participants will be able to:

1. Understand how the RAM API works from looking up the correct function to navigating the exposed database
2. Learn how to create dynamo custom nodes using zero touch C#
3. Learn how to create, modify, and extract data from existing RAM SS Models using the API with C# dynamo custom nodes
4. Learn how to share your Tekla API knowledge with contractors and become their hero

THIS CLASS WILL NOT FOCUS ON HOW TO PASS MODELS BETWEEN REVIT AND RAM SS
WILL SHOW C# PROGRAMMING BUT NOT FOR EVERY EXAMPLE

This is You
AT YOUR OFFICE
After this Class

“The RAM
Superhero”



This CLASS uses the following versions

RAM 15*

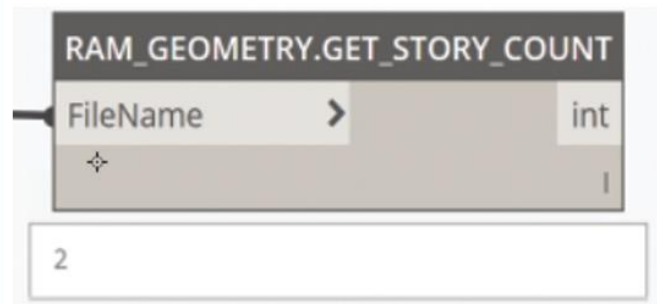
ETABS 17

Dynamo 1.3.3 OR Dynamo 2.X OK

Revit 2019 OR Revit 2020 OK

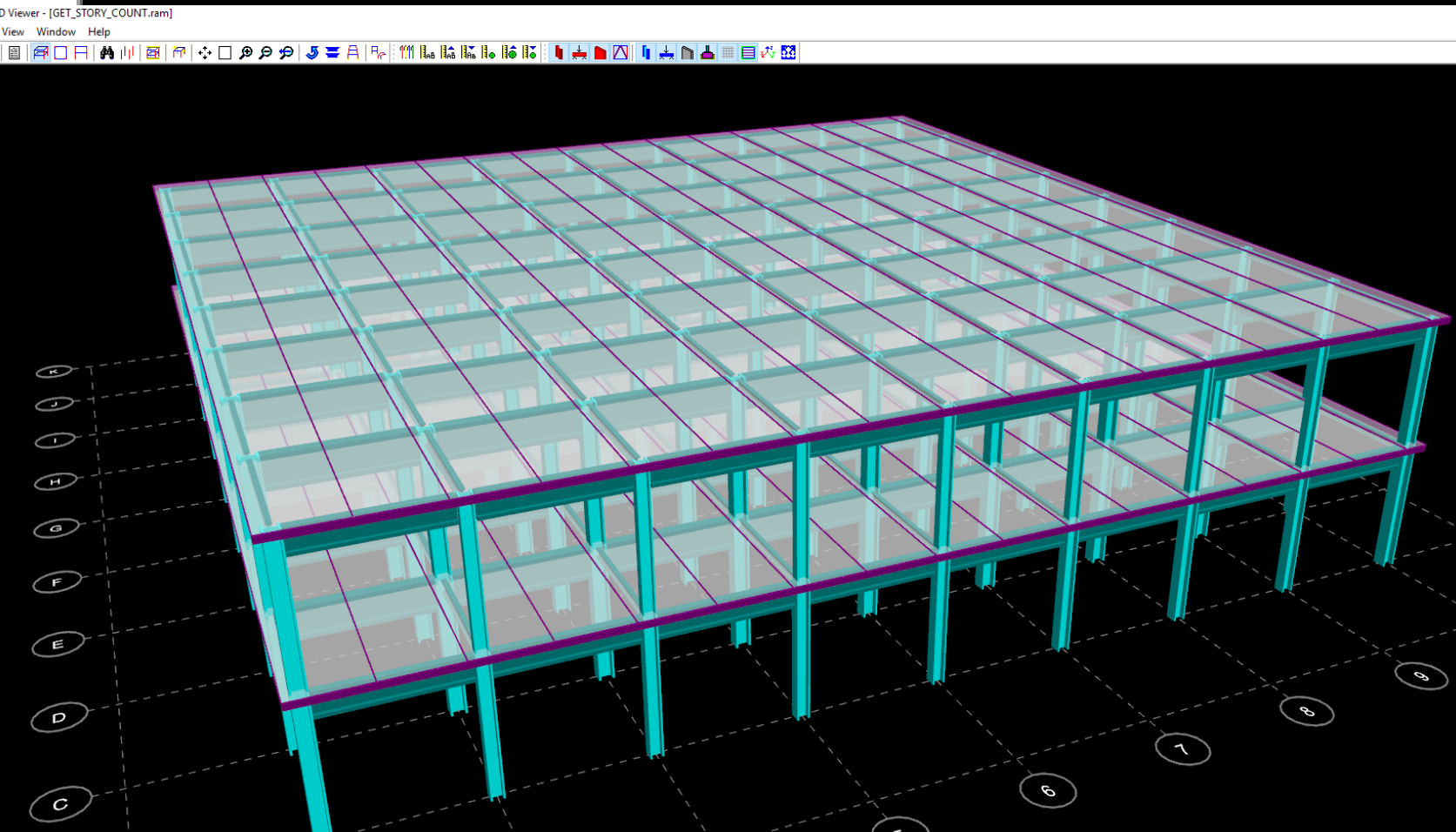
Dynamo Simplex Package 2019.X

Note: RAM 16 out next month on SIMPLEX

GET NUMBER OF STORIES IN RAM USING DYNAMO USING RAM API AND ZERO TOUCH C#		
<pre>using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks; using RAMDATAACCESSLib; using Autodesk.DesignScript.Runtime; using Autodesk.DesignScript.Geometry; namespace DYNAMO_FOR_RAM_SIMPLE { public class RAM_GEOMETRY { private RAM_GEOMETRY() { } public static int GET_STORY_COUNT(string FileName) { RamDataAccess1 RAMDataAccess = new RAMDATAACCESSLib.RamDataAccess1(); RAMDATAACCESSLib.IDBIO1 IDBI = (RAMDATAACCESSLib.IDBIO1) RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IDBIO1 INT); RAMDATAACCESSLib.IModel IModel = (RAMDATAACCESSLib.IModel) RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IModel_INT); IDBI.LoadDataBase2(FileName, "1"); IStories My_stories = IModel.GetStories(); int My_story_count = My_stories.GetCount(); IDBI.CloseDatabase(); return My_story_count; } } }</pre>		DYNAMO NODE
<p>TYPE BEING SENT TO OUTPUT PORT</p> <p>NAME OF NODE</p> <p>INPUT PORT</p> <p>VARIABLES THAT ACCESSES THE "IMODEL" OBJECT-THAT HAS ALL THE DATA NEEDED</p> <p>OPENS DATADASE VIA FILE</p> <p>GETS "ISTORY" OBJECT</p> <p>GETS NUMBER OF STORIES FROM "ISTORY"</p> <p>CLOSES DATADASE</p> <p>SENDS NUMBER OF STORIES TO DYNAMO OUTPUT PORT</p>		ZERO TOUCH C# RAM API CODE
<p>IDBIO1 Philosophy: This interface is for performing functions on the database as a whole. This includes Loading and Saving. It also includes changing the database name. It does not include inquiries or specific data manipulation.</p> <p>IModel The <u>IModel</u> interface is the highest level in the hierarchy of interfaces. It represents the RAM Structural System model. Entities such as section definitions and floor types that are global to the model can be accessed through this interface. Other entities, such as decks or beams, which belong to a specific floor type, can be accessed through this interface using the entity's unique ID.</p> <p>GetStories ([out, retval] IStories** pplStories) Gets the collection of all stories in the model. Parameters Pointer to an IStories collection interface that represents all stories in the model pplStories</p> <p>GetCount ([out, retval] long* plCount) Gets the number of stories in the collection. Parameters plCount Number of stories in the collector</p>		RAM API MANUAL
STEP 1: OPEN VISUAL STUDIO FOLDER "GET_NUM_STORIES" OPEN SLN FILE - BUILD SLN STEP 2: OPEN DYNAMO AND LOAD DLL AND PLACE NODE STEP 3: SELECT THE RAM FILE VIA FILE PATH NODE AND WATCH THE RESULTS NOTE: RAM DOES NOT NEED TO BE OPEN, AND NOTE RELATIONSHIP WITH API DOC AND CODE		STEPS & NOTES

CHEATSHEET FORMAT!

ZERO TOUCH EXAMPLE (starter)



RAM BASICS

RAM Structural System is an integrated three-dimensional static and dynamic structural analysis and design program owned by Bentley. With Very rigid modelling rules and w/ superior post processing!

operation failed.

The diagram shows a "Code Block" containing the text "NEWFLOORTYPE";. A blue arrow points from this code block to a yellow box representing a RAM variable named "RAM_GEOMETRY_WRITE.SET_FLOOR_TYPE". This variable has two fields: "FloorTypeName" of type "int" and "FileName" of type ">". Below this variable box is a white box labeled "null". A red arrow points from the text "AS WRITE- THE RAM USED THE" to the "FileName" field.

ANY ERROR THAT OCCURS WILL CREATE A
(THE NAME AS RAM FILE).USR AND IT IS
LOCATED NEXT TO THE ORIGINAL RAM FILE
IT'S CASE LOOK FOR
E "WRITE_TO_RAM_FLOORTYPE.USR"
SIMPLY DELETE IT!

Name

- DYNAMO_FOR_RAM_SIMPLE
- Interop.RAMDATAACCESSLib.dll
- RAM DataAccess Developers Guide.pdf
- WRITE_TO_RAM_FLOORTYPE.backup
- WRITE_TO_RAM_FLOORTYPE.dyn
- WRITE_TO_RAM_FLOORTYPE.png
- WRITE_TO_RAM_FLOORTYPE.psd
- WRITE_TO_RAM_FLOORTYPE.rss
- write_to_ram_floortype.usr
- WRITE_TO_RAM_FLOORTYPE

[illegible]

LOCAL MODEL ORGANIZATION

```
public static List<Autodesk.DesignScript.Geometry.Line> GET_RAM_COL_CL(string FileName, int In_Story_Count
```

```

RAMDataAccess RAMDataAccess = new RAMDataAccessLib.RAMDataAccess();
RAMDataAccessLib.IDB001 IDB1 = (RAMDataAccessLib.IDB001)
RAMDataAccess.getInterfacePointerByEnum(EINTERFACEACES.IDB001_INT);
RAMDataAccessLib.Model Model = (RAMDataAccessLib.Model)
RAMDataAccess.getInterfacePointerByEnum(EINTERFACEACES.IDB001_INT);
//OPEN
IDB1.OpenDatabase(filename, "1");
IDB1ories My_storys = IDB1.getStories();
int My_story_count = My_storys.getCount();
IDB1ories My_story = My_storys.getItem(InStory_count);
IDB1ories My_Columns = My_story.getItemColumns();
int Column_Count = My_Columns.getCount();
Coordinate P1 = new Coordinate();
Coordinate P2 = new Coordinate();
List<Autodesk.DesignScript.Geometry.Line> ListLine = new List<Autodesk.DesignScript.Geometry.Line>();
//create loop hereuntil all count
for (int i = 0; i < Column_Count; i = i + 1)
{
    My_story.getItemColumns().getItem(i).getCoordinates(ref P1, ref P2);
    double P1x = P1.dL0x;
    double P1y = P1.dL0y;
    double P1z = P1.dL0z;
    double P2x = P2.dL0x;
    double P2y = P2.dL0y;
    double P2z = P2.dL0z;
    Autodesk.DesignScript.Geometry.Point P01 =
Autodesk.DesignScript.Geometry.Point.ByCoordinates(P1x, P1y, P1z);
Autodesk.DesignScript.Geometry.Point P02 =
Autodesk.DesignScript.Geometry.Point.ByCoordinates(P2x, P2y, P2z);
Autodesk.DesignScript.Geometry.Line lLine =
Autodesk.DesignScript.Geometry.Line.ByStartPointEndPoint(P01, P02);
ListLine.Add(lLine);
}

```

RAM_GEOMETRY_READ_WRITE.GET_RAM_BM_CL

RAM_GEOMETRY.GET_STORY_COUNT

FileName

>

int

+

I

2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using RAMDATAACCESSLib;
using Autodesk.DesignScript.Runtime;
using Autodesk.DesignScript.Geometry;
```

```
namespace DYNAMO FOR RAM SIMPLE
```

```

public class RAM_GEOMETRY
{
    private RAM_GEOMETRY()
    {
    }

    public static int GET_STORY_COUNT(string FileName)
    {
        RamDataAccess1 RAMDataAccess = new RAMDATAACCESSLib.RamDataAccess1();
        RAMDATAACCESSLib.IDBI01 IDBI = (RAMDATAACCESSLib.IDBI01)
        RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IDBI01 INT);
        RAMDATAACCESSLib.IModel IModel = (RAMDATAACCESSLib.IModel)
        RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IModel_INT);

        IDBI.LoadDatabase2(FileName, "1");

        IStories My_stories = IModel.GetStories();
        int My_story_count = My_stories.GetCount();
        IStory My_story = My_stories.Getat(In_story_Count);
        IColumns My_Columns = My_story.GetColumns();
        int Column_Count = My_Columns.GetCount();

        IDBI.CloseDatabase();

        return Column_Count;
    }
}

```

MAKE LIST OF STORIES
GETS ALL COL PER STORY

Code Block
X 0..X-1.; >

RAM_GEOMETRY.GET
FileName
In_story_Count

VARIABLES THAT ACCESSES THE "IMODEL" OBJECT-THAT HAS ALL THE DATA NEEDED

OPENS DATABASE VIA FILE

GETS "STORY" OBJECT

GETS NUMBER OF STORIES FROM "STORY"

GETS "STORY" OBJECT FROM STORIES AT INDEX

CLOSES DATABASE

GETS ICOLUMNS OBJECT AT EACH ISTORY OBJECT

SENDS DYNAMO OUTPUT PORT

IDBIO1
Philosophy: This interface is for performing functions on the database as a whole. This includes Loading and Saving. It also includes changing the database name. It does not include inquiries or specific data manipulation.

IModel

The [IModel](#) interface is the highest level in the hierarchy of interfaces. It represents the RAM Structural System model. Entities such as section definitions and floor types that are global to the model can be accessed through this interface. Other entities, such as decks or beams, which belong to a specific floor type, can be accessed through this interface using the entity's unique ID.

GetStories ([out, retval] IStories** pplStories) Gets the collection of all stories in the model. Parameters Pointer to an IStories collection in pplStories that represents all stories in the m
--

GetCount ([out, retval] long* pICount)
Gets the number of stories in the collection.

Parameters
pICount Number of stories in the collection

ZERO TOUCH C#P

API MANUAL

RAM API BASICS

int storyCount = 2;
Console.WriteLine(storyCount);

IMPLEMENT

TYPE BEING SENT

RAM_GEOMETRY.GET_STORY_COUNT

FileName int

2

The diagram illustrates the following code segments and their functions:

- Static Initialization:**

```
static int GET_STORY_COUNT(string FileName)
```
- Database Access Setup:**

```

RAMDataAccess RAMDataAccess = new RAMDataAccessLib.RamDataAccess();
RAMDataAccessLib.IDBIO1 IDBIO1 = (RAMDataAccessLib.IDBIO1)
    RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IDBIO1 INT);
RAMDataAccessLib.IModel IModel = (RAMDataAccessLib.IModel)
    RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IModel_INT);
    
```
- Database Connection:**

```
IDBIO1.LoadDataBase2(FileName, "1");
```
- Story Retrieval:**

```
IStories My_stories = IModel.GetStories();
```
- Count Retrieval:**

```
int My_story_count = My_stories.GetCount();
```
- Database Closure:**

```
IDBIO1.CloseDatabase();
```
- Return Statement:**

```
return My_story_count;
```

Annotations:

- VARIABLES THAT ACCESSES THE "IMODEL" OBJECT-THAT HAS ALL THE DATA NEEDED** (points to RAMDataAccess and IModel)
- OPENS DATABASE VIA FILE** (points to IDBIO1.LoadDataBase2)
- GETS "ISTORY" OBJECT** (points to IModel.GetStories)
- GETS NUMBER OF STORIES FROM "ISTORY"** (points to My_stories.GetCount)
- CLOSES DATABASE** (points to IDBIO1.CloseDatabase)
- SENDS NUMBER OF STORIES TO DYNAMO OUTPUT PORT** (points to return My_story_count)

IDBIO1
Philosophy: This interface is for performing functions on the database as a whole. This includes Loading and Saving. It also includes changing the database name. It does not include inquiries or specific data manipulation.

IModel
The ***IModel*** interface is the highest level in the hierarchy of interfaces. It represents the RAM Structural System model. Entities such as section definitions and floor types that are global to the model can be accessed through this interface. Other entities, such as decks or beams, which belong to a specific floor type, can be accessed through this interface using the entity's unique ID.

GetStories ([out, retval] IStories** pplStories)
Gets the collection of all stories in the model.
Parameters: *pplStories* - an IStories collection to store the retrieved

C# RAM API CODE AND DYNAMIC NODE

CODE AND DYNAMO NODE

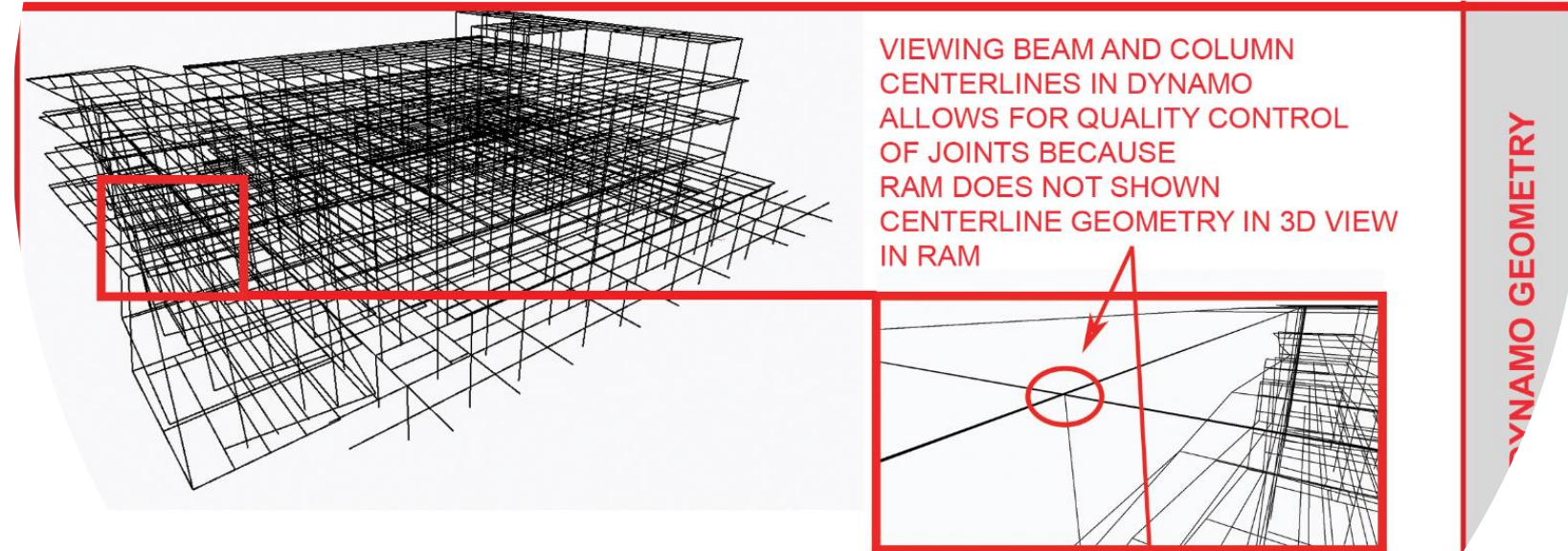
1000

1
2
3
4
5
6
7
8
9
10
11
12
13
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
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
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
84

MANUAL

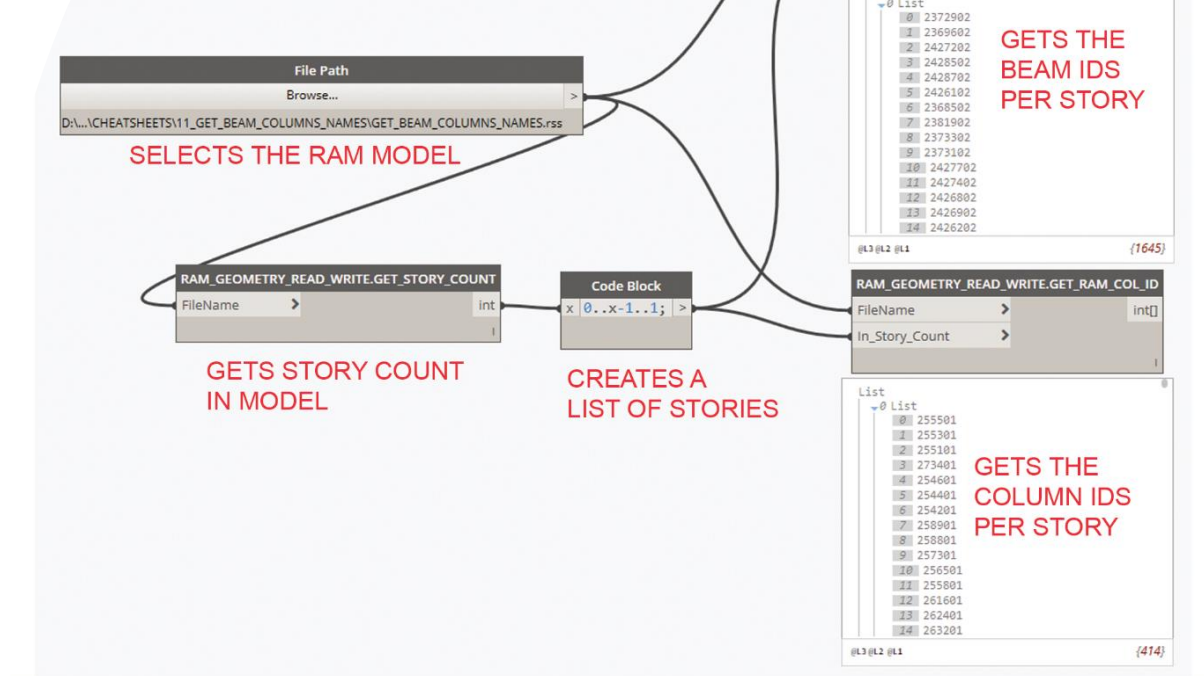
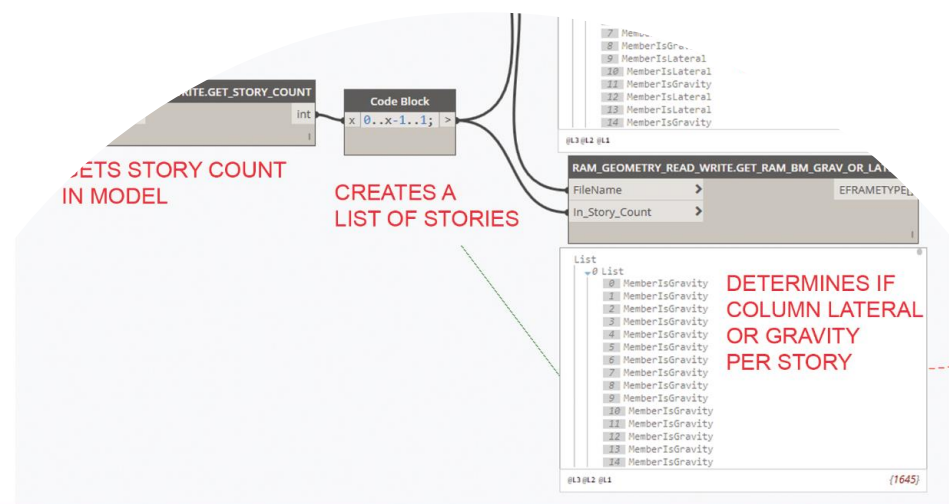
STEP 1: OPEN VISUAL STUDIO FOLDER "GET_NUM_COLUMNS" OPEN SLN FILE - BUILD SLN
STEP 2: OPEN DYNAMO AND LOAD DLL AND PLACE NODES

STEP 3: SELECT THE RAM FILE VIA FILE PATH NODE AND WATCH THE RESULTS

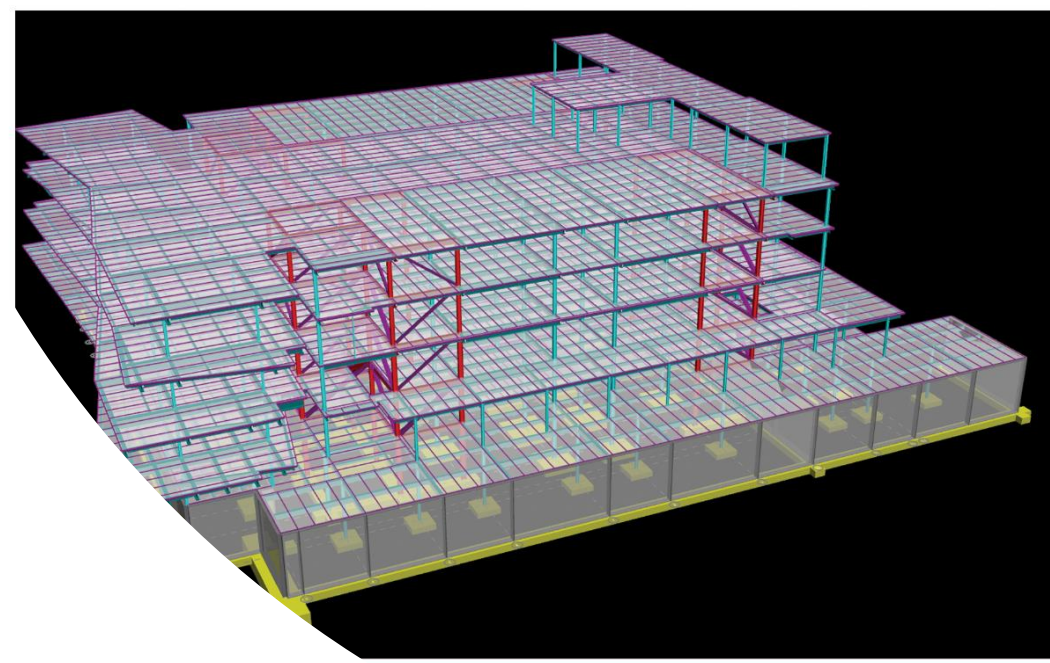
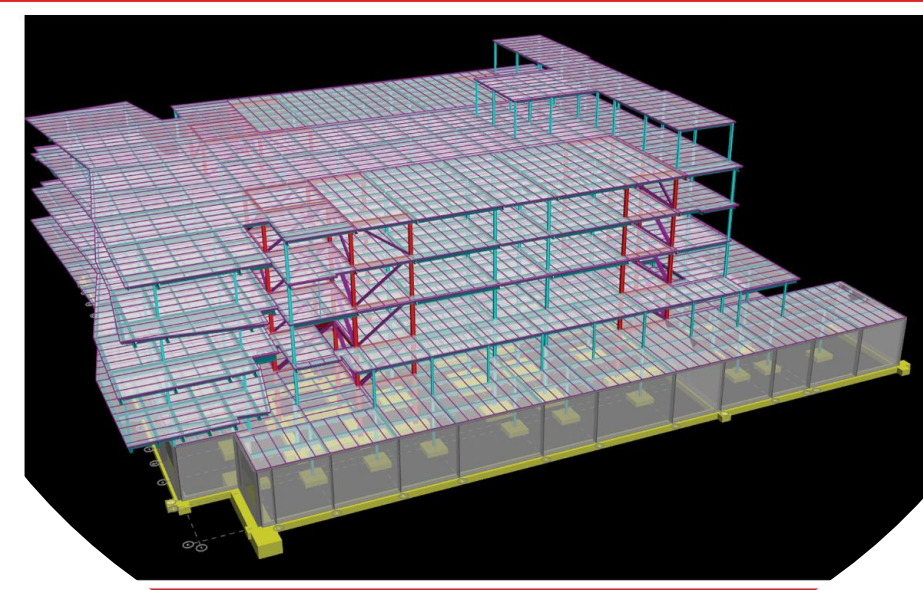
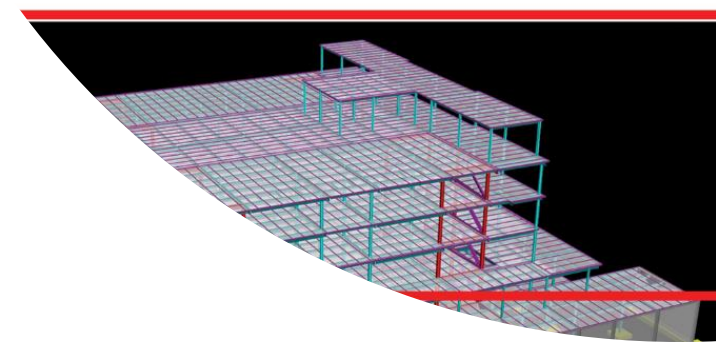


VIEWS BEAM AND COLUMN CENTERLINES IN DYNAMO
ALLOWS FOR QUALITY CONTROL OF JOINTS BECAUSE RAM DOES NOT SHOWN CENTERLINE GEOMETRY IN 3D VIEW IN RAM

DYNAMO GEOMETRY

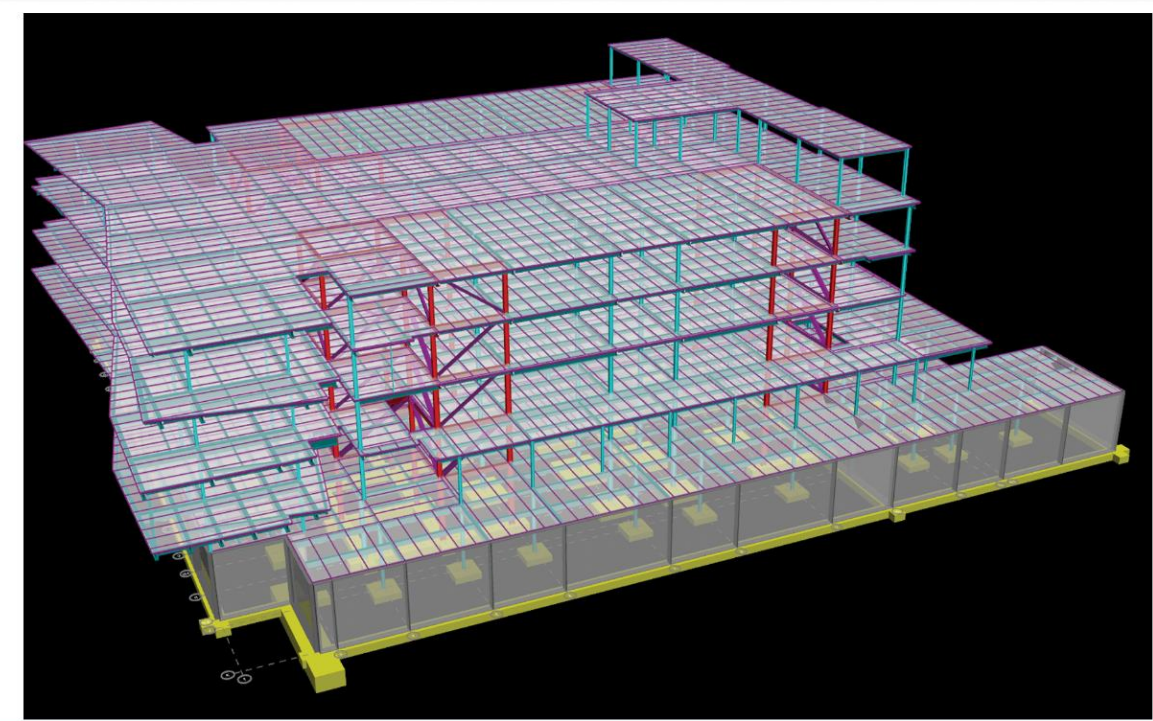
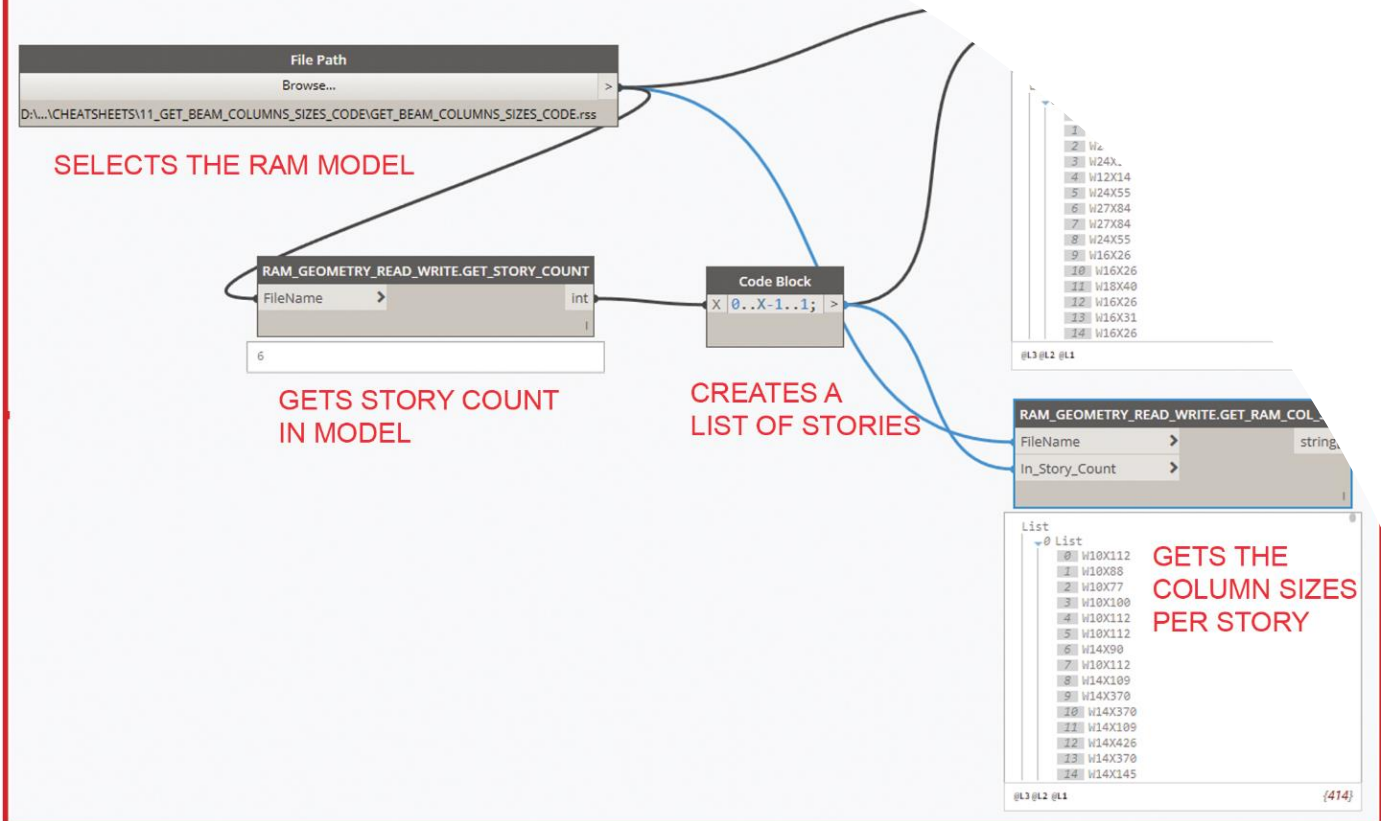


DYNAMO NODES



RAM MODEL

GET BEAM AND COLUMN IDS PER STORY USING DYNAMO



RAM MODEL

RAM API READ DATA

STEP 1: OPEN VISUAL STUDIO FOLDER "GET_BEAM_COLUMN_SIZES" OPEN SLN FILE
STEP 2: OPEN DYNAMO AND LOAD DLL AND PLACE NODE ADD STORIES AS A LIST
STEP 3: SELECT THE RAM FILE VIA FILE PATH NODE AND WATCH THE RESULTS
NOTE: RAM DOES NOT NEED TO BE OPEN. SEE SIMPLEX PACKAGE AND RAM API MANUAL

STEPS & NOTES

Open RAM api

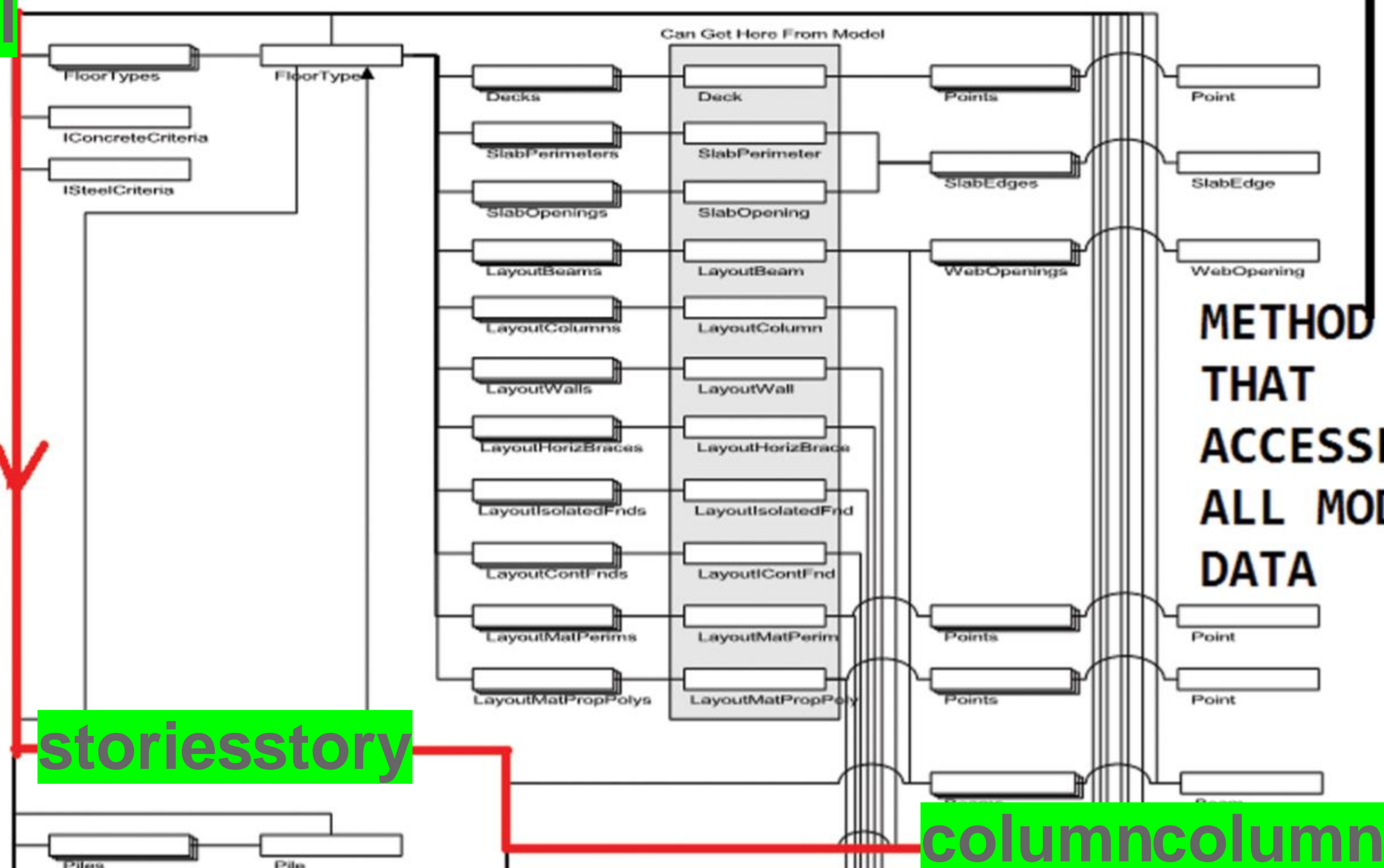
```
RAMDATAACCESSLib.IModel IModel =  
RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IModel_INT);
```

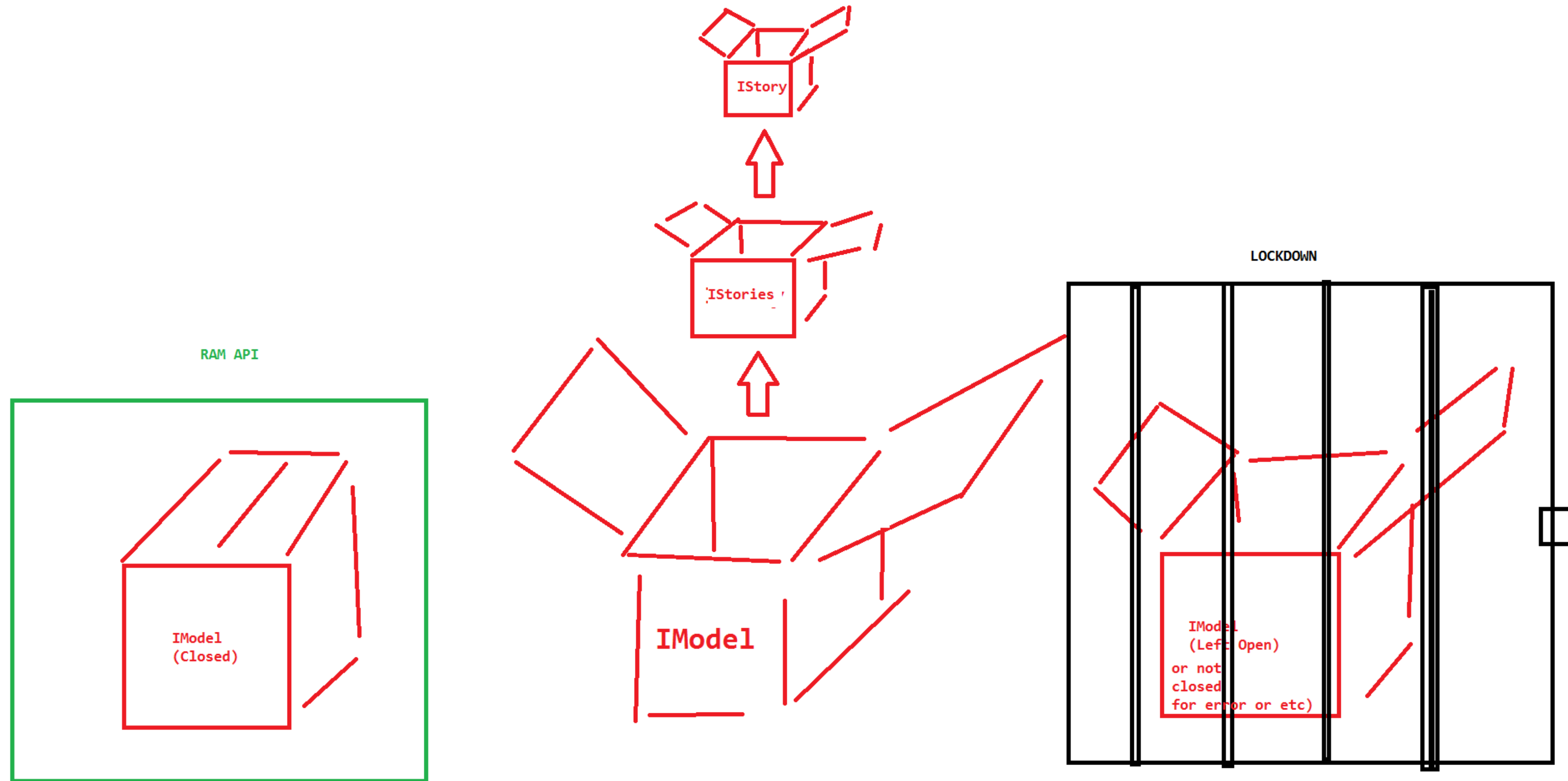
model

RAM API
ORG
CHART

storiesstory

columncolumn props





Open box analogy

GET NUMBER OF STORIES IN RAM USING DYNAMO
USING RAM API AND ZERO TOUCH C#

g System;

g System.Collections.Generic;

g System.Linq;

g System.Text;

g System.Threading.Tasks;

g RAMDATAACCESSLib;

g Autodesk.DesignScript.Runtime;

g Autodesk.DesignScript.Geometry;

space DYNAMO_FOR_RAM_SIMPLE

public class RAM_GEOMETRY

{

private RAM_GEOMETRY()

{

}

public static int GET_STORY_COUNT(string FileName)

{

RamDataAccess1 RAMDataAccess = new RAMDATAACCESSLib.RamDataAccess1();

RAMDATAACCESSLib.IDBIO1 IDBI = (RAMDATAACCESSLib.IDBIO1

RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IDBIO1 INT);

RAMDATAACCESSLib.IModel IModel = (RAMDATAACCESSLib.IModel)

RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IModel_INT);

IDBI.LoadDataBase2(FileName, "1");

IStories My_stories = IModel.GetStories();

int My_story_count = My_stories.GetCount();

IDBI.CloseDatabase();

return My_story_count;

}

}

RAM_GEOMETRY.GET_STORY_COUNT

FileName > int

2

TYPE BEING SENT TO OUTPUT PORT

NAME OF NODE

INPUT PORT

VARIABLES THAT ACCESSES THE "IMODEL" OBJECT-THAT HAS ALL THE DATA NEEDED

OPENS DATADASE VIA FILE

GETS "ISTORY" OBJECT

GETS NUMBER OF STORIES FROM "ISTORY" OBJECT

CLOSES DATADASE

SENDS NUMBER OF STORIES TO DYNAMO OUTPUT PORT

IDBIO1
Philosophy: This interface is for performing functions on the database as a whole. This includes Loading and Saving. It also includes changing the database name. It does not include inquiries or specific data manipulation.

IModel
The [IModel](#) interface is the highest level in the hierarchy of interfaces. It represents the RAM Structural System model. Entities such as section definitions and floor types that are global to the model can be accessed through this interface. Other entities, such as decks or beams, which belong to a specific floor type, can be accessed through this interface using the entity's unique ID.

GetStories ([out, retval] IStories** pplStories)
Gets the collection of all stories in the model.
Parameters Pointer to an IStories collection interface that represents all stories in the model

GetCount ([out, retval] long* plCount)
Gets the number of stories in the collection.
Parameters plCount Number of stories in the collector

TEP 1: OPEN VISUAL STUDIO FOLDER "GET_NUM_STORIES" OPEN SLN FILE - BUILD SLN

TEP 2: OPEN DYNAMO AND LOAD DLL AND PLACE NODE

TEP 3: SELECT THE RAM FILE VIA FILE PATH NODE AND WATCH THE RESULTS

OOTE: RAM DOES NOT NEED TO BE OPEN, AND NOTE RELATIONSHIP WITH API DOC AND CODE

GET NUMBER OF COLUMNS/STORY IN RAM VIA DYNAMO USING RAM API AND ZERO TOUCH C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using RAMDATAACCESSLib;
using Autodesk.DesignScript.Runtime;
using Autodesk.DesignScript.Geometry;
```

```
space DYNAMO_FOR_RAM_SIMPLE

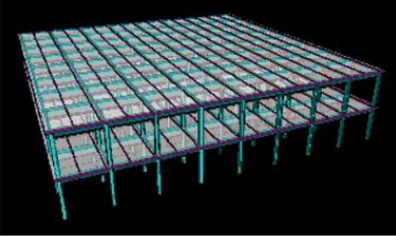
public class RAM_GEOMETRY
{
    private RAM_GEOMETRY()
    {
    }

    public static int GET_STORY_COUNT(string FileName)
    {
        RamDataAccess1 RAMDataAccess = new RAMDATAACCESSLib.RamDataAccess1();
        RAMDATAACCESSLib.IDBI01 IDBI = (RAMDATAACCESSLib.IDBI01)
        RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IDBI01 INT);
        RAMDATAACCESSLib.IModel IModel = (RAMDATAACCESSLib.IModel)
        RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IModel_INT);

        IDBI.LoadDataBase2(FileName, "1");
        IStories My_stories = IModel.GetStories();
        int My_story_count = My_stories.GetCount();
        IStory My_Story = My_stories.GetAt(In_Story_Count);
        IColumns My_Columns = My_Story.GetColumns();
        int Column_Count = My_Columns.GetCount();

        IDBI.CloseDatabase();

        return Column_Count;
    }
}
```



GETS STORY COUNT
MAKE LIST OF STORIES
GETS ALL COL PER STORY

File Path
Browse...
D:\...\CHEATSHEETS\08_GET_NUM_COLUMNS\GET_NUM_COLUMNS.rss

RAM_GEOMETRY.GET_STORY_COUNT
FileName int

Code Block
X 0..X-1..1; >

RAM_GEOMETRY.GET_COLUMN_COUNT
FileName int
In_Story_Count List
0 81
1 81

VARIABLES THAT ACCESSES THE "IMODEL" OBJECT-THAT HAS ALL THE DATA NEEDED

OPENS DATADASE VIA FILE

GETS "ISTORY" OBJECT

GETS NUMBER OF STORIES FROM "ISTORY"

GETS "ISTORY" OBJECT FROM ISTORIES AT INDEX

CLOSES DATADASE

SENDS DYNAMO OUTPUT PORT

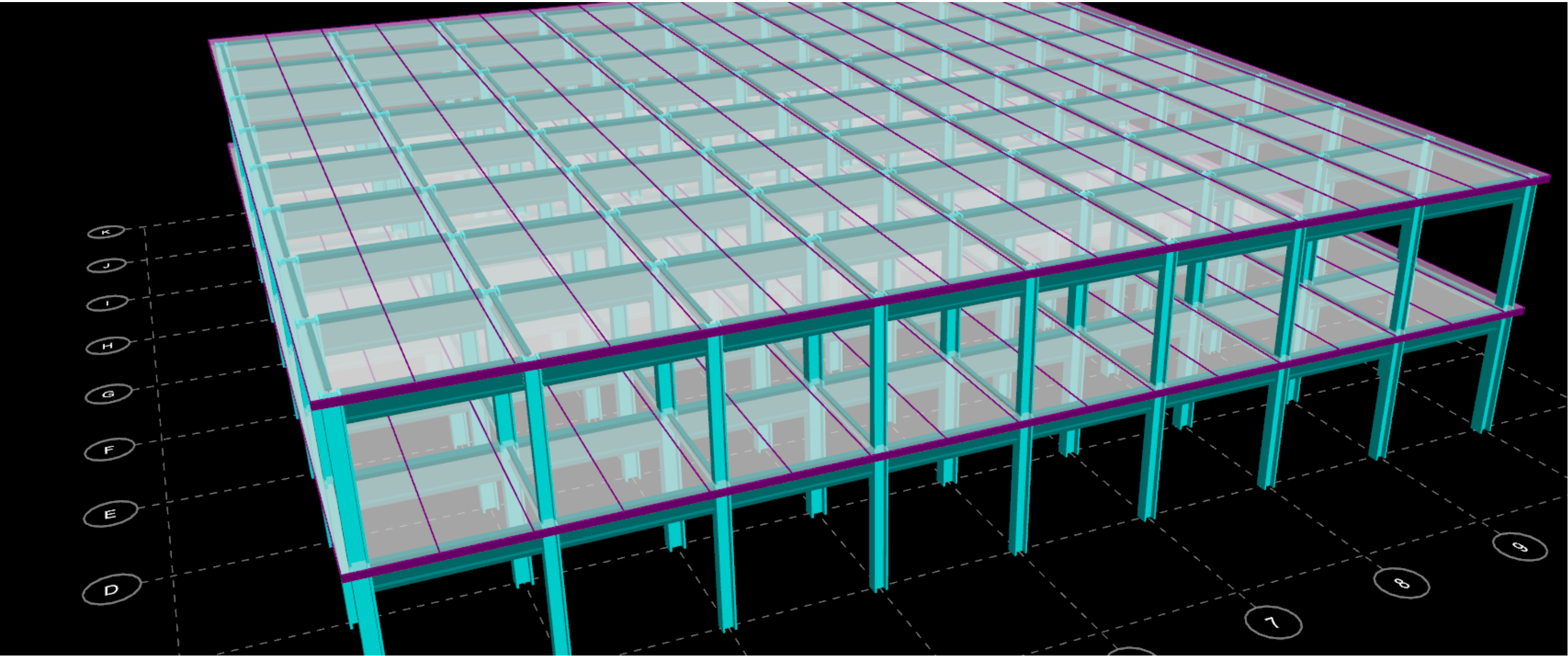
GETS ICOLUMNS OBJECT AT EACH ISTORY OBJECT

IDBI01
Philosophy: This interface is for performing functions on the database as a whole. This includes Loading and Saving. It also includes changing the database name. It does not include inquiries or specific data manipulation.

IModel
The IModel interface is the highest level in the hierarchy of interfaces. It represents the RAM Structural System model. Entities such as section definitions and floor types that are global to the model can be accessed through this interface. Other entities, such as decks or beams, which belong to a specific floor type, can be accessed through this interface using the entity's unique ID.

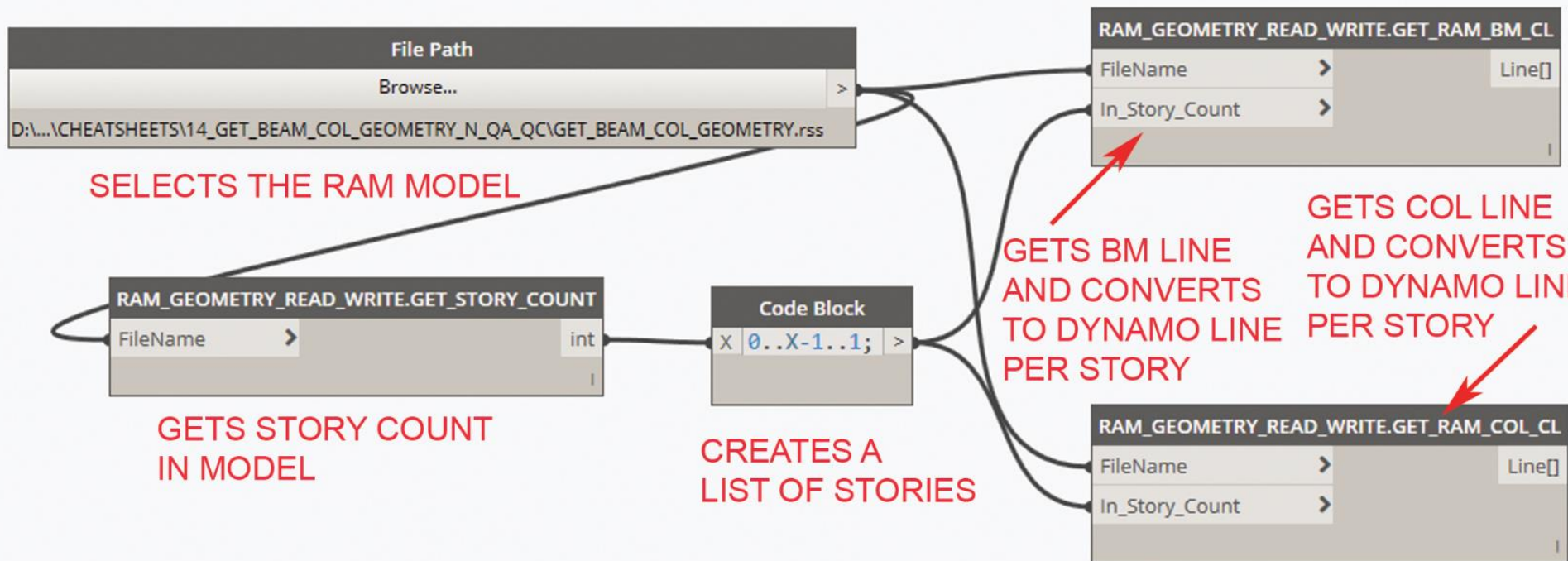
GetStories ([out, retval] IStories** pplStories)
Gets the collection of all stories in the model.
Parameters Pointer to an IStories collection interface that represents all stories in the model

GetCount ([out, retval] long* plCount)
Gets the number of stories in the collection.
Parameters plCount Number of stories in the collector

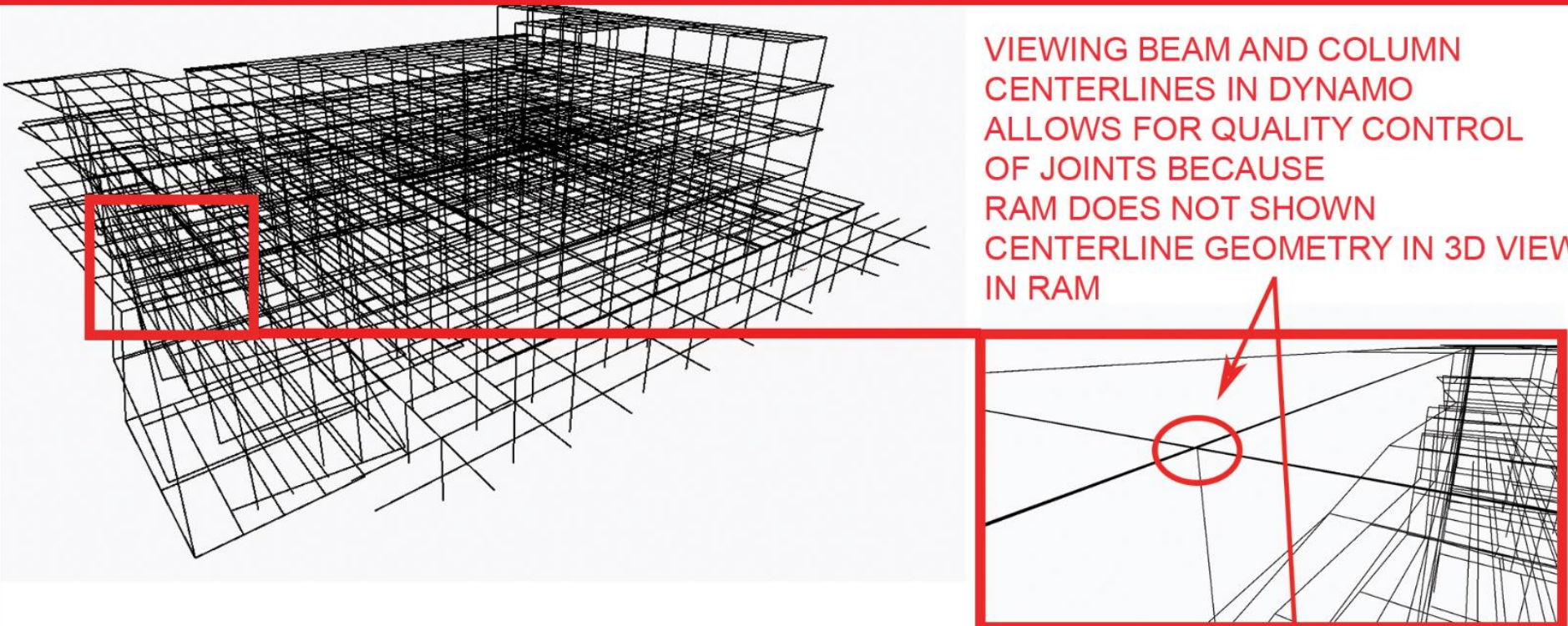


- TEP 1: OPEN VISUAL STUDIO FOLDER "GET_NUM_COLUMNS" OPEN SLN FILE - BUILD SLN
- TEP 2: OPEN DYNAMO AND LOAD DLL AND PLACE NODES
- TEP 3: SELECT THE RAM FILE VIA FILE PATH NODE AND WATCH THE RESULTS
- OTE: RAM DOES NOT NEED TO BE OPEN, AND NOTE RELATIONSHIP WITH API DOC AND CODE

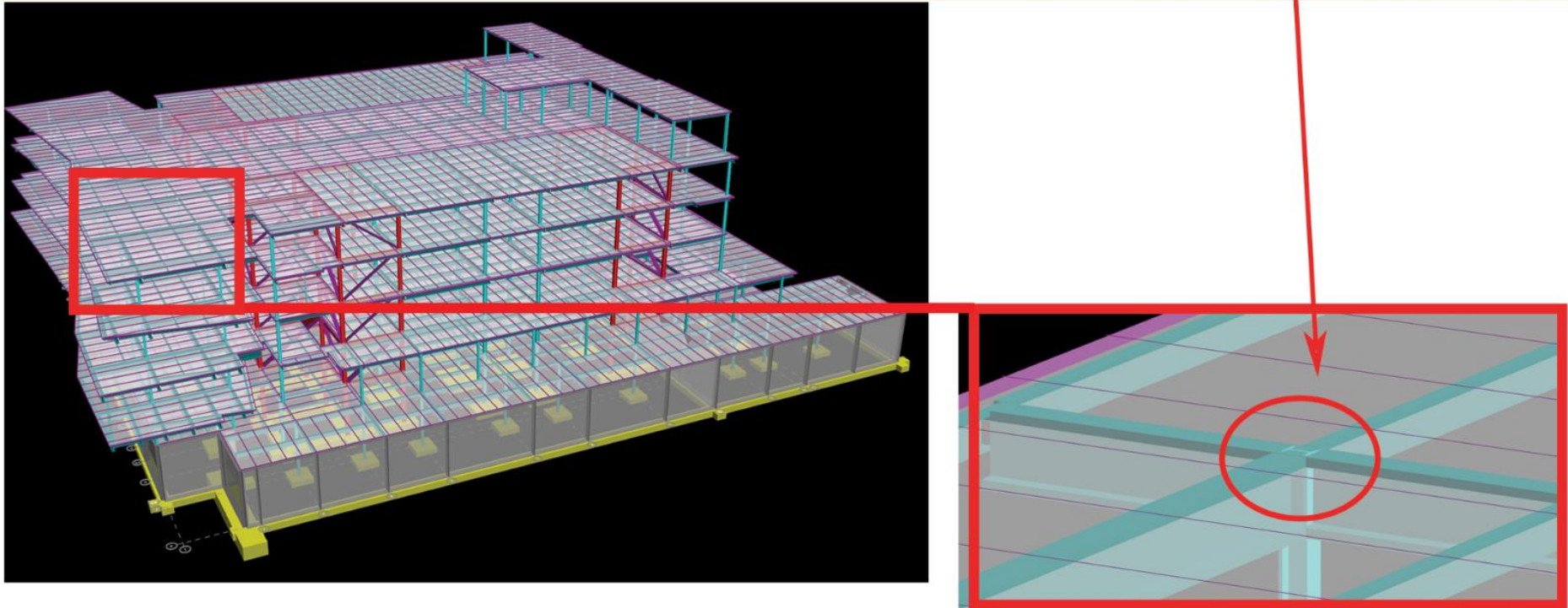
GET COLUMN AND BEAM CENTERLINES FROM RAM
AND CONVERT TO DYNAMO LINES



DYNAMO NODES



DYNAMO GEOMETRY



RAM MODEL

- STEP 1: OPEN VISUAL STUDIO FOLDER "GET_BEAM_COL_GEOMETRY" OPEN SLN FILE
STEP 2: OPEN DYNAMO AND LOAD DLL AND PLACE NODE ADD STORIES AS A LIST
STEP 3: SELECT THE RAM FILE VIA FILE PATH NODE AND WATCH THE RESULTS
NOTE: RAM DOES NOT NEED TO BE OPEN. SEE SIMPLEX PACKAGE AND RAM API MANUAL

STEPS & NOTES

WRITES HORIZONTAL BRACE
VIA DYNAMO USING RAM API

```
static int CREATE_RAM_STEEL_BRACE(int FloorIndex, string FileName, double StartSupportX, double StartSupportY, double StartSupportZ, double EndSupportX, double EndSupportY, double EndSupportZ)
{
    RamDataAccess1 RAMDataAccess = new RAMDATAACCESSLib.RamDataAccess1();
    RAMDATAACCESSLib.IDBIO1 IDBI = (RAMDATAACCESSLib.IDBIO1)
    RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IDBIO1_INT);
    RAMDATAACCESSLib.IModel IModel = (RAMDATAACCESSLib.IModel)
    RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IModel_INT);
    IDBI.LoadDataBase2(FileName, "1");

    IFloorTypes My_floortypes = IModel.GetFloorTypes();
    IFloorType My_floortype = My_floortypes.GetAt(FloorIndex);
    EMATERIALTYPES My_BmMaterial = EMATERIALTYPES.ESteelMat;

    ILayoutBrace My_LayoutBrace = My_floortype.
    GetLayoutHorizBraces().Add(My_BmMaterial,
    StartSupportX, StartSupportY,
    StartSupportZ, EndSupportX, EndSupportY, EndSupportZ);

    IDBI.SaveDatabase();
    IDBI.CloseDatabase();
    int My_New_Brace_ID = My_LayoutBrace.LUID;
    return My_New_Brace_ID;
}
```

GETS IMODEL OBJECT FROM RAM

GETS FLOORTYPES OBJECT FROM IMODEL

GETS SINGLE FLOORTYPE AT INDEX FROM INPUT PORT

GETS STEEL MATERIAL OBJECT

GETS ILAYOUTBRACE OBJECT FROM SINGLE FLOORTYPE AND CREATE A SINGLE ILAYOUTBRACE OBJECT THAT DEFINES A SINGLE BRACE

GETS NEW BEAM AND RETURNS ITS ID

WRITES NEW BRACE TO FLOOR (INDEXED) AND OUTPUTS NEW BRACE ID
NOTE: CODE SHOWN ABOVE

OPEN FOLDER "WRITE_BRACE" OPEN SLN FILE
OPEN DYNAMO AND LOAD DLL AND PLACE NODE ADD STOP
SELECT THE RAM FILE IN FOLDER VIA FILE PATH NODE
DOES NOT NEED TO BE OPEN. ALSO SEE SIMP

```
RAM_GEOMETRY_READ_WRITE.GET_RAM_COL_CL
{
    IDBI.IDBIO1 IDBI = (RAMDATAACCESSLib.IDBIO1)
    RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IDBIO1_INT);
    RAMDATAACCESSLib.IModel IModel = (RAMDATAACCESSLib.IModel)
    RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IModel_INT);
    IDBI.LoadDataBase2(FileName, "1");

    IFloorTypes My_floortypes = IModel.GetFloorTypes();
    IFloorType My_floortype = My_floortypes.GetAt(FloorIndex);
    EMATERIALTYPES My_BmMaterial = EMATERIALTYPES.ESteelMat;

    ILayoutBeam My_LayoutBeam = My_floortype.GetLayoutBeams()
    .Add(My_BmMaterial, StartSupportX, StartSupportY,
    StartSupportZ, EndSupportX, EndSupportY, EndSupportZ);

    //CLOSE
    IDBI.SaveDatabase();
    IDBI.CloseDatabase();
    int My_New_Beam_ID = My_LayoutBeam.LUID;
    return My_New_Beam_ID;
}
```

GETS IMODEL OBJECT FROM RAM

GETS FLOORTYPES OBJECT FROM IMODEL

GETS SINGLE FLOORTYPE AT INDEX FROM INPUT PORT

GETS STEEL MATERIAL OBJECT

GETS ILAYOUTBEAMS OBJECT FROM SINGLE FLOORTYPE AND CREATE A SINGLE ILAYOUTBEAM OBJECT THAT DEFINES A SINGLE BEAM

GETS NEW BEAM AND RETURNS ITS ID

GETS CENTER LINES OF COLUMNS AND PLACES POINTS AT ENDS

WRITES NEW BEAMS TO FLOOR (INDEXED) AND OUTPUTS NEW BEAM ID
NOTE: CODE SHOWN ABOVE

```
static int CREATE_RAM_STEEL_COL(int FloorIndex, string FileName, double XX, double YY, double ZTop, double ZBot)
{
    RamDataAccess1 RAMDataAccess = new RAMDATAACCESSLib.RamDataAccess1();
    RAMDATAACCESSLib.IDBIO1 IDBI = (RAMDATAACCESSLib.IDBIO1)
    RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IDBIO1_INT);
    RAMDATAACCESSLib.IModel IModel = (RAMDATAACCESSLib.IModel)
    RAMDataAccess.GetInterfacePointerByEnum(EINTERFACES.IModel_INT);
    IDBI.LoadDataBase2(FileName, "1");

    IFloorTypes My_floortypes = IModel.GetFloorTypes();
    IFloorType My_floortype = My_floortypes.GetAt(FloorIndex);
    EMATERIALTYPES My_ColMaterial = EMATERIALTYPES.ESteelMat;

    ILayoutColumn My_LayoutColumn = My_floortype.GetLayoutColumns()
    .Add(My_ColMaterial, XX, YY, ZTop, ZBot);

    //CLOSE
    IDBI.SaveDatabase();
    IDBI.CloseDatabase();
    int My_New_Col_ID = My_LayoutColumn.LUID;
    return My_New_Col_ID;
}
```

GETS IMODEL OBJECT FROM RAM

GETS FLOORTYPES OBJECT FROM IMODEL

GETS SINGLE FLOORTYPE AT INDEX FROM INPUT PORT

GETS STEEL MATERIAL OBJECT

GETS ILAYOUTCOLUMNS OBJECT FROM SINGLE FLOORTYPE AND CREATE A SINGLE ILAYOUTCOLUMN OBJECT THAT DEFINES A SINGLE COLUMN

GETS NEW COLUMN AND RETURNS ITS ID

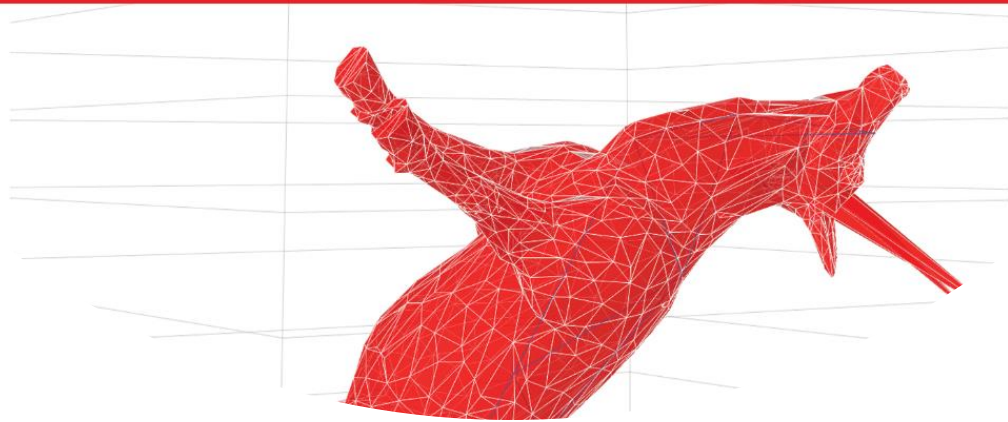
FLOOR INDEX (FLOOR 1=0 FLOOR 2=1 ETC)

WRITES NEW COLUMNS TO FLOOR (INDEXED)
NOTE: CODE SHOWN ABOVE

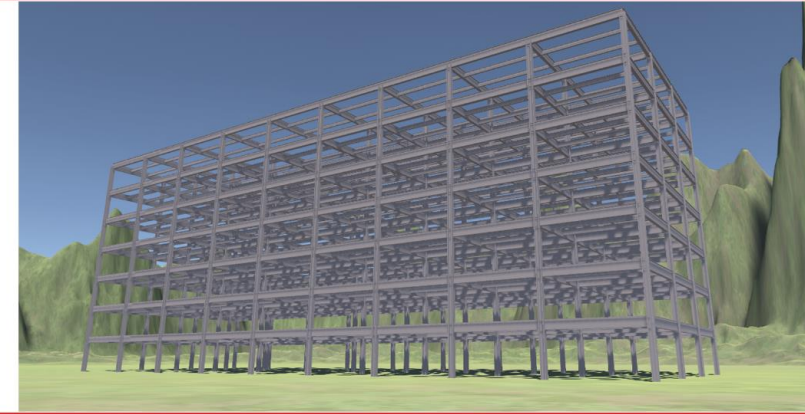
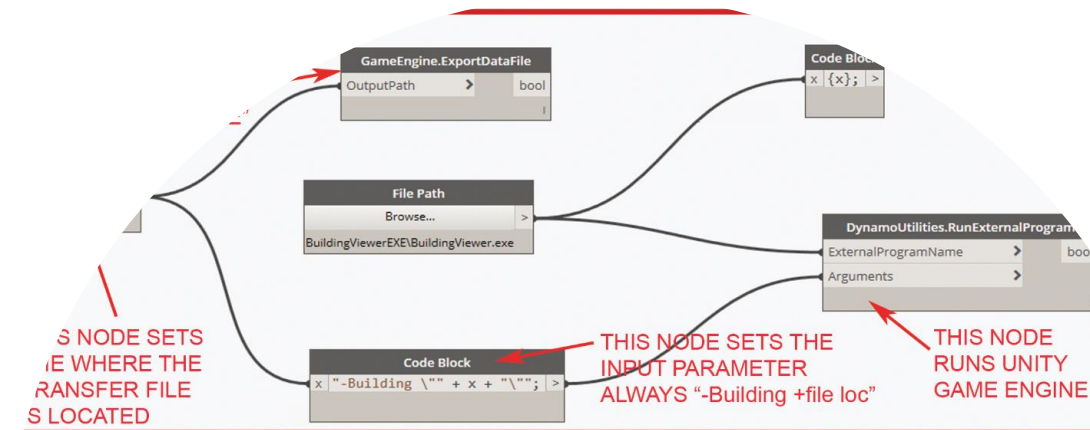
WRITES NEW COLUMNS TO FLOOR (INDEXED)
NOTE: CODE SHOWN ABOVE

OPEN SLN FILE
PLACE NODE ADD STORIES AS A LIST
WITH NODE AND WATCH THE RESULTS
SEE DYNAMO PACKAGE AND RAM API MANUAL

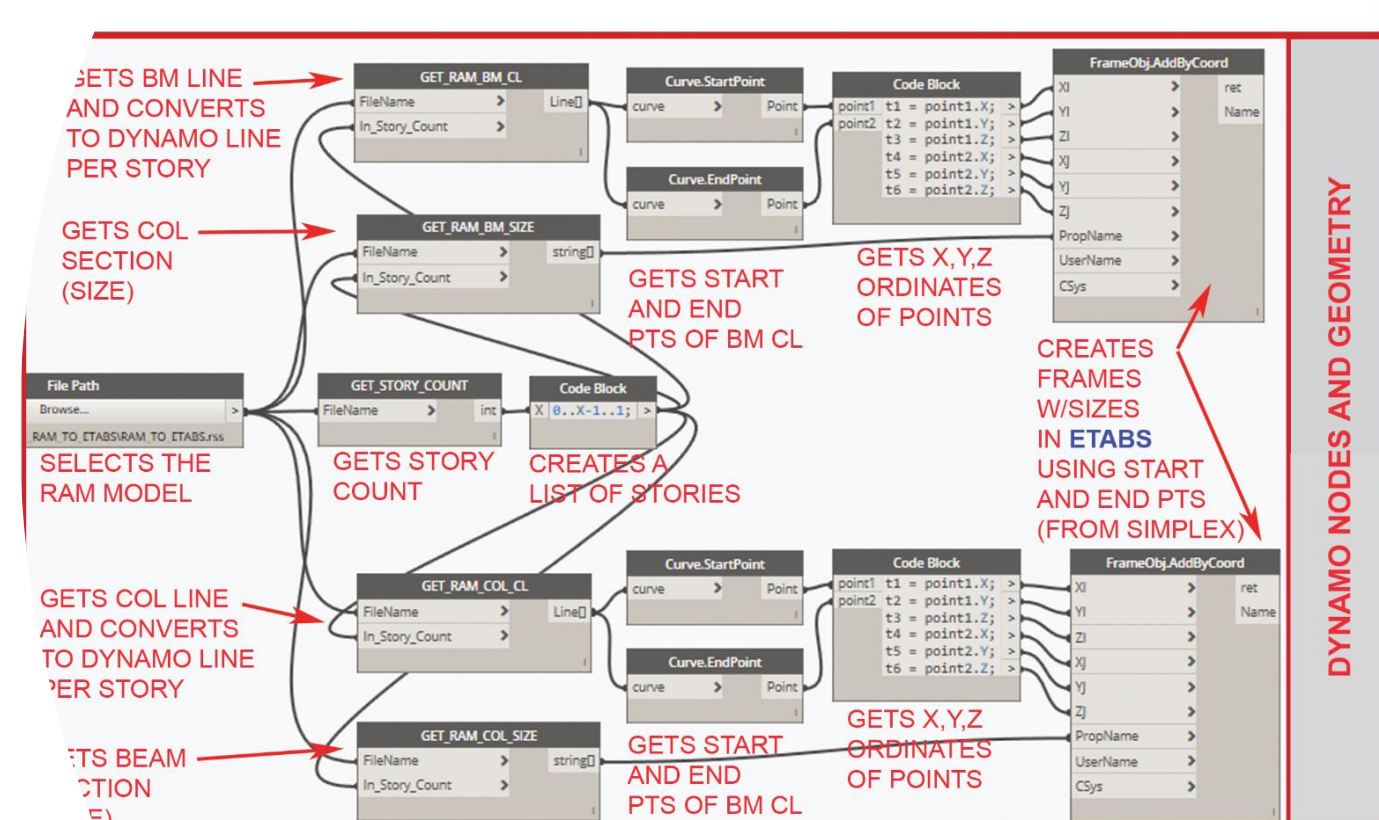
RAM API WRITE DATA



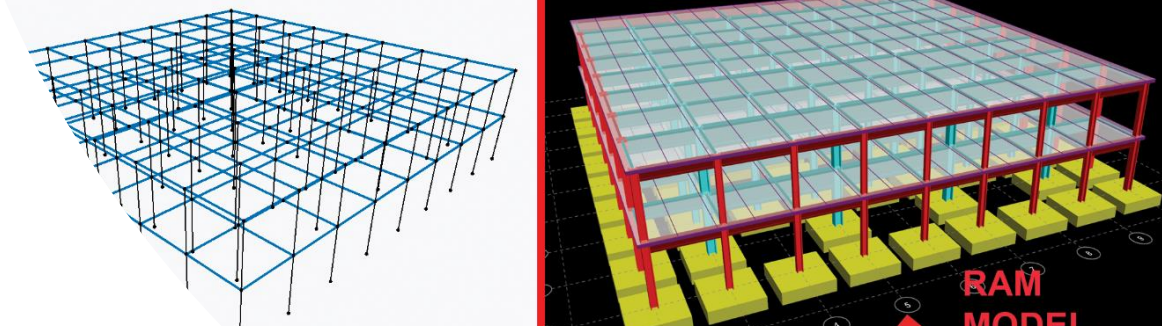
DYNAMO GEOMETRY



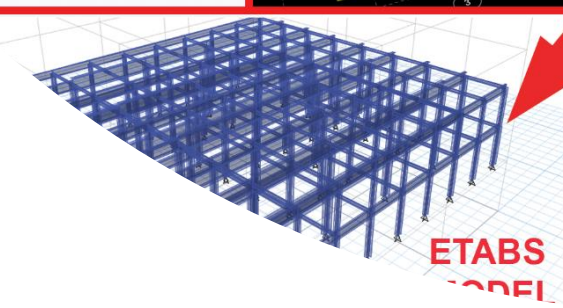
GAME ENGINE GEOMETRY



DYNAMO NODES AND GEOMETRY



RAM MODEL



ETABS MODEL

BEAMS NODES FROM SIMPLEX

GETS SELECTED ELEMENTS IN ETABS MODEL

GETS BM LINE AND CONVERTS TO DYNAMO LINE

GETS START AND END PTS OF BM CL

GETS X,Y,Z ORDINATES OF POINTS

SELECTS THE RAM MODEL

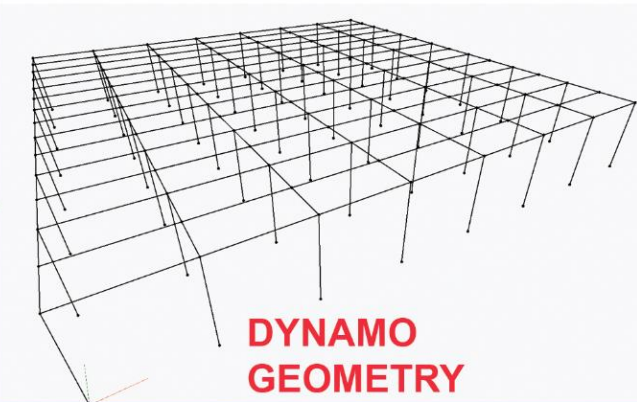
CREATES BEAMS AND COLUMNS IN RAM PER FLOORTYPE

COLUMNS NODES FROM SIMPLEX

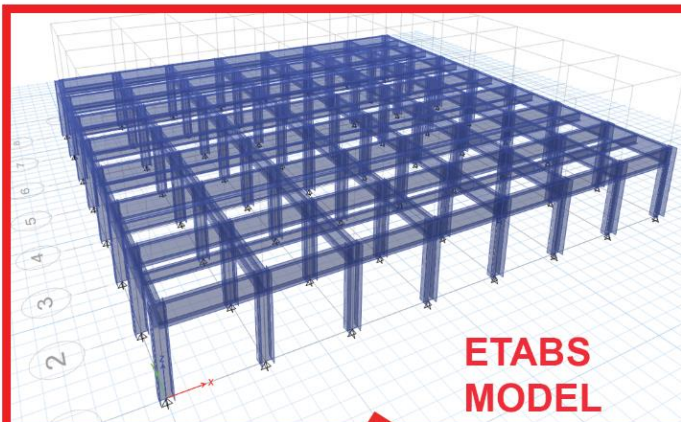
GETS START AND END PTS OF CL

GETS X,Y,Z

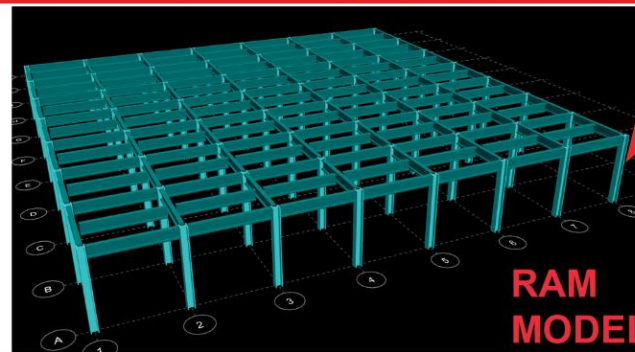
RAM_GEOMETRY_READ_WRITE.CREATE_RAM_STEEL_COLUMN



DYNAMO GEOMETRY



ETABS MODEL



RAM MODEL

ETABS MODEL

MODEL

RAM API TRANSFER DATA TO OTHER DATABASES

HELLO ETABS AND BEYOND

STEP 1: OPEN FOLDER "ETABS_TO_RAM" OPEN SLN FILE, BUILD SOLUTION
STEP 2: OPEN ETABS FILE "ETABS_TO_RAM_START.EDB"
STEP 3: OPEN DYNAMO, LOAD RECENTLY BUILT DLL, PLACE NODES AS SHOWN (MAKE SURE TO HAVE SIMPLEX INSTALL)
STEP 4: SELECT THE RAM FILE VIA FILE PATH NODE AND WATCH THE RESULTS

This is You After this Class





AUTODESK®

Make anything™

Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2019 Autodesk. All rights reserved.

