OED to ktools mapping algorithm

Introduction

This document outlines how the new Reinsurance format **Open Exposure Data** (OED) is mapped to the Oasis ktools module **fm_calc. [todo]?**

Overview of algorithm

Step 1 – define reinsurance grouping

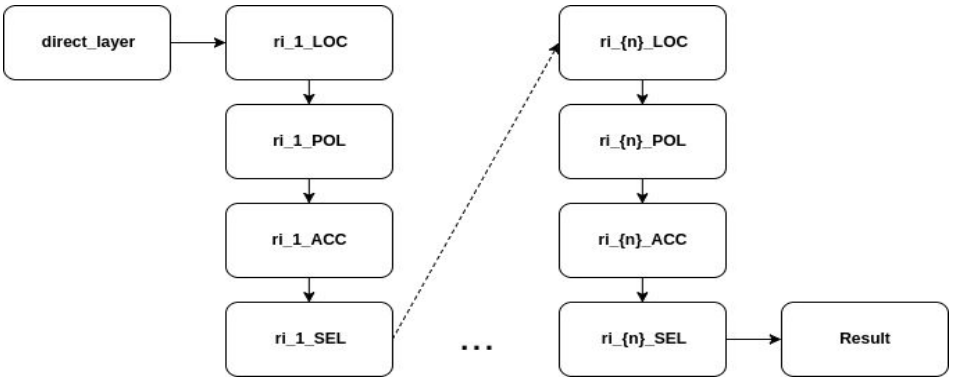
a) Read in the OED files, which are:

ri_info.csv - defines the type of reinsurance, **InuringPriority** and terms & conditions.
ri_scope.csv – rows are matched to an entry in the ri_info.csv file based on the **ReinsNumber** value. This sets the **RiskLevel** and defines which subset of the exposures to apply the Reinsurance calculation rules to.
accounts.csv – Exposure data for accounts
location.csv - Exposure data for locations

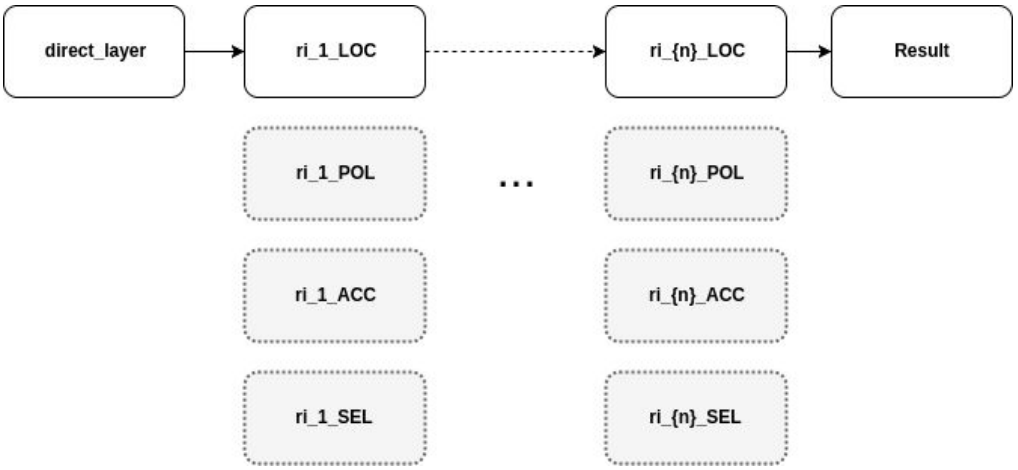
b) The reinsurance groupings are created based on the combination of **InuringPriority** and **RiskLevel.** The reinsurance groupings form the stages of the calculation.

The Oasis fm-calc module is run for each grouping of **ri_{InuringPriority}_{RiskLevel}** in the order of InuringPriority **[1 .. n]** and RiskLevel **[Location → Policy → Account → Portfolio]** .

For each grouping the **net_loss** output is used as the **pre_loss** for the next stage in execution order.



If a **Risklevel** or **InuringPriority** is missing from the order of execution, then **net_loss** is routed to the next valid grouping. If the only **RiskLevel** defined is **LOC**, location level, then net_loss from ri_1_LOC is used in **pre_loss** for ri_2_LOC.



Step 2 – Creating the Oasis files

Each reinsurance grouping requires its own set of oasis files for the calculation.

```
For each ri_group in reinsurence_groups:
     # Create Node Tree
     # Select nodes and apply calculation rules
     generate_oasis_structures()
     # Write out ktools files
     write_oasis_files()
```

Node Trees for Multiple_FAC:

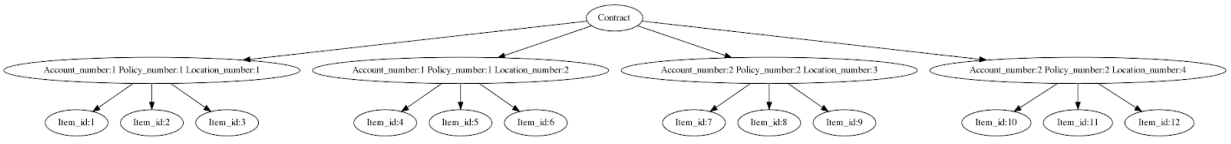
In this instance we have a single InuringPriority of '**1**' and three RiskLevels '**LOC, POL, and ACC**' giving us three reinsurance groupings **[ri_1_LOC → ri_1_POL → ri_1_ACC]**

These structures are analogous to the financial module programme hierarchy represented in the fm_programme file. For more information see Financial Module.

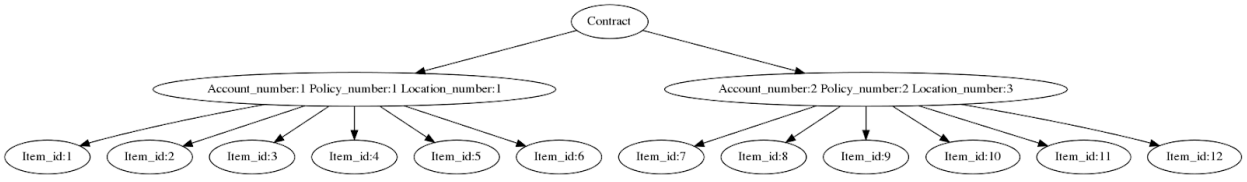
Each node is aggregated to the parent node which has is own aggregation ID at that level of the tree.

Note that these are the complete trees based on the OED files provided. In order to apply the calculation rules we need to iterate over these structures and check which rules should be applied based on the reinsurance files **ri_info.csv** and **ri_scope.csv**

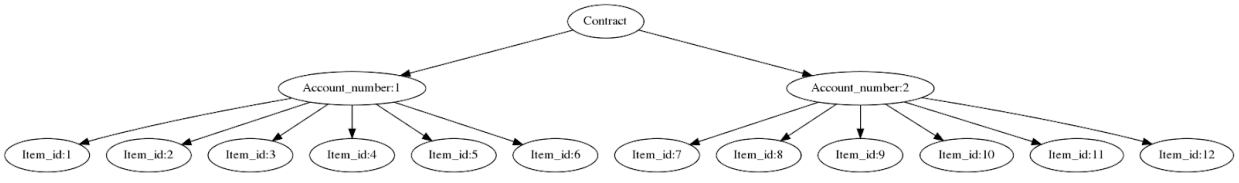
Location (ri_1_LOC)



Policy (ri_1_POL)



Account (ri_1_ACC)



Applying the FM Calculation Rules

The pseudo code below is a summary of how rules are selected for each inuring/risk_level grouping. For clarity, most of the logic is abstracted behind “functions” that are not present in the actual code. This is to give an overview of what's happening without getting lost in the details. The full source code is available here.

```
For each ri_info_row in InuringPriority:

    # Layers are based on the ReinsNumber used for each row
    # in ri_info.csv
    if (ri_info_row.ReinsNumber is not  prev_ReinsNumber)
        create_new_ktools_layer()

    # Depending on the ReinsType we apply FM calcRules to
    # different sections of the node tree
    # surplus_share, cat_xl, facultative.. etc
    select_rules_to_apply(ri_info_row.ReinsType)

    # For the top node in the tree add rules which will apply with all
    # groupings in the treaty
    if (ri_info_row.OccLimit or
        ri_info_row.CededPercent or
        ri_info_row.PlacementPercent):
         apply_calcRule(23) # OCCURRENCE_LIMIT_AND_SHARE
    else:
         apply_calcRule(PASSTHROUGH_RULE)

    # Iterate over the nodes and check if each is within scope
    # Based on ri_scope.csv
    For each level_2_node in tree:
        if matches(current_node, ri_scope):
             apply_calcRule(24) # OCCURRENCE_CATASTROPHE_EXCESS_OF_LOSS
        else:
             apply_calcRule(NO_LOSS_RULE)
```

Explanation of Pseudocode functions

create_new_ktools_layer(): In the oasis ktools fmcalc module, layers are used to apply multiple calculation rules to the same agg_id. For more details on how this works see the layers subsection of the ktools documentation.

select_rules_to_apply(ri_info_row.ReinsType): The logic used to create the ktools files depends on the reinsurance type used in the **ri_info.csv** file. In the code base these are stored in functions for each type of reinsurance.

- add_fac_profiles()
- add_per_risk_profiles()
- add_surplus_share_profiles()
- add_quota_share_profiles()
- add_cat_xl_profiles()

apply_calcRule(n): Add an fmcalc rule for the current node in the iteration. The majority of FM calculation rules can be covered using CalcRule 24 or `**occurrence catastrophe excess of loss**`. If part of the node tree needs to be excluded from the Treaty, or the complete tree, then we apply a `No Loss` rule of CalcRule 14 without a limit applied, this passes the **pre_loss** value to the output as is.

matches(current_node, ri_scope): Check if a node should be included or excluded from the reinsurance calculation. In the example below for `**multiple_FAC**` in the grouping for **ri_1_LOC** only accountNumber == 1 is in scope.

Generation of oasis files

Multiple_FAC Example

At Inuring Priority **1** and RiskLevel **LOC** there are two rows from the **ri_info** file used each with a single scope, matched using the same ReinsNumber.

ri_info.csv:

ReinsNumber	ReinsLayerNumber	InuringPriority	LocationNumb	ReinsType
1	1	1	1	FAC
2	1	1	2	FAC
3	1	1	3	FAC
4	1	1	4	FAC

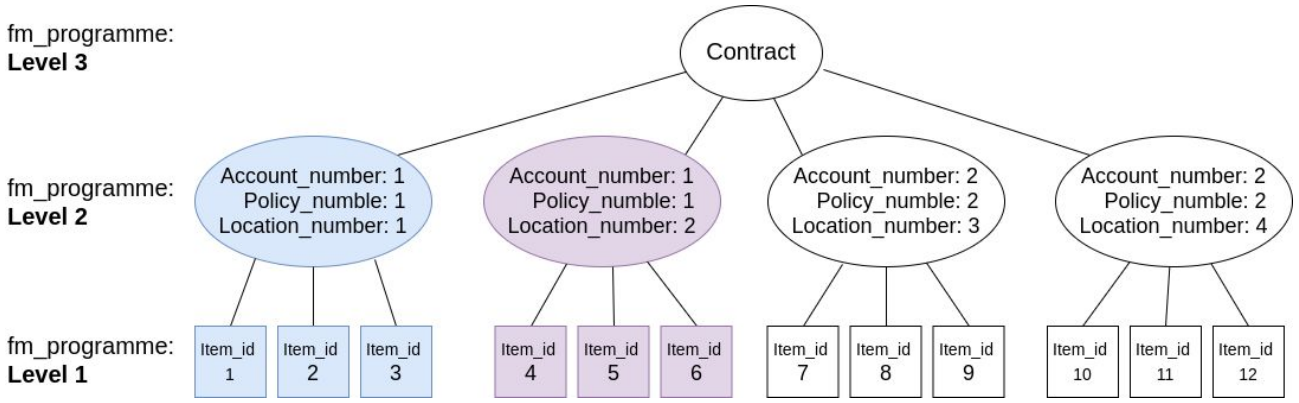
ri_scope.csv

ReinsNumber	RiskLevel	PortfolioNumber	AccountNumber	PolicyNumber
1	LOC	1	1	1
2	LOC	1	1	1
3	POL	1	1	1
4	ACC	1	2	1

When iterating over the 2nd level of the tree, each node is checked against the reinsurance input files using

(ri_info.LocationNumber, ri_scope.AccountNumer, ri_scope.PolicyNumber)

which gives us two matches for account Numbers [1,2]



Note: The item_id's map to coverage_id's for each location node, with the following Total Insured Values.

item_id	coverage_id	TIV	LocationNumber
1	1	1000	1
2	2	500	1
3	3	500	1
4	4	1000	2
5	5	500	2
6	6	500	2
7	7	1000	3
8	8	500	3
9	9	500	3
10	10	1000	4
11	11	500	4
12	12	500	4

fm_programme.csv

from_agg_id	level_id	to_agg_id
1	2	1
2	2	1
3	2	1
4	2	1
1	1	1
2	1	1
3	1	1
4	1	2
5	1	2
6	1	2
7	1	3
8	1	3
9	1	3
10	1	4
11	1	4
12	1	4

fm_policytc.csv

layer_id	level_id	agg_id	profile_id
1	2	1	2
1	1	1	3
1	1	2	1
1	1	3	1
1	1	4	1
2	2	1	2
2	1	1	1
2	1	2	4
2	1	3	1
2	1	4	1

fm_profile.csv

profile_id	calcrule_id	deductible1	deductible2	deductible3	attachment	limit	share1	share2	share3
1	14	0	0	0	0	0	0	0	0
2	12	0	0	0	0	0	0	0	0
3	24	0	0	0	0	10	1	1	1
4	24	0	0	0	0	10	1	1	1

Multiple_SS Example

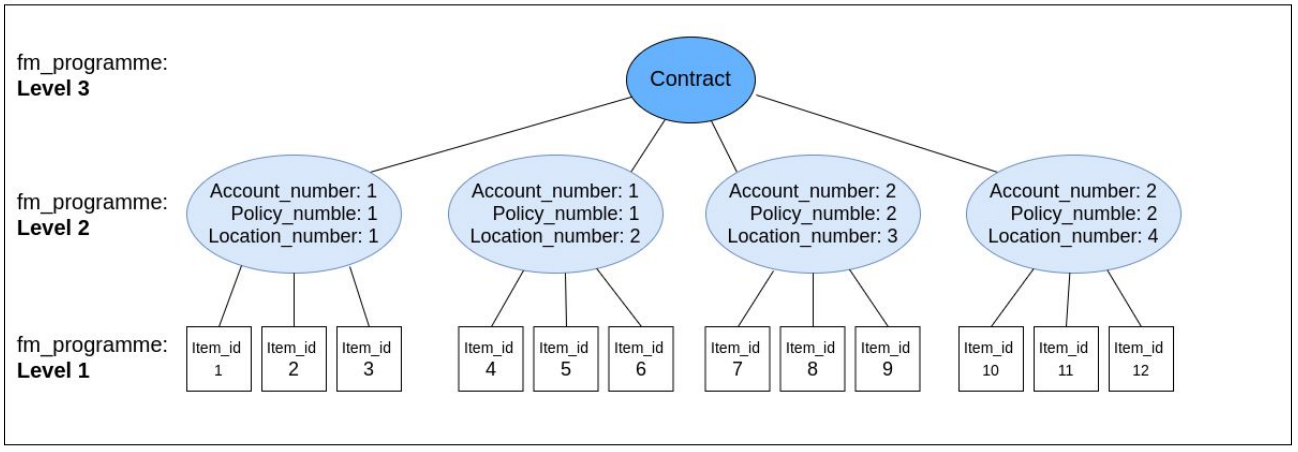
ri_info.csv:

ReinsNumber	ReinsLayerNumber	CededPercent	InuringPriority	ReinsType	PlacementPercent
1	1	1	1	SS	1
2	2	1	1	SS	1

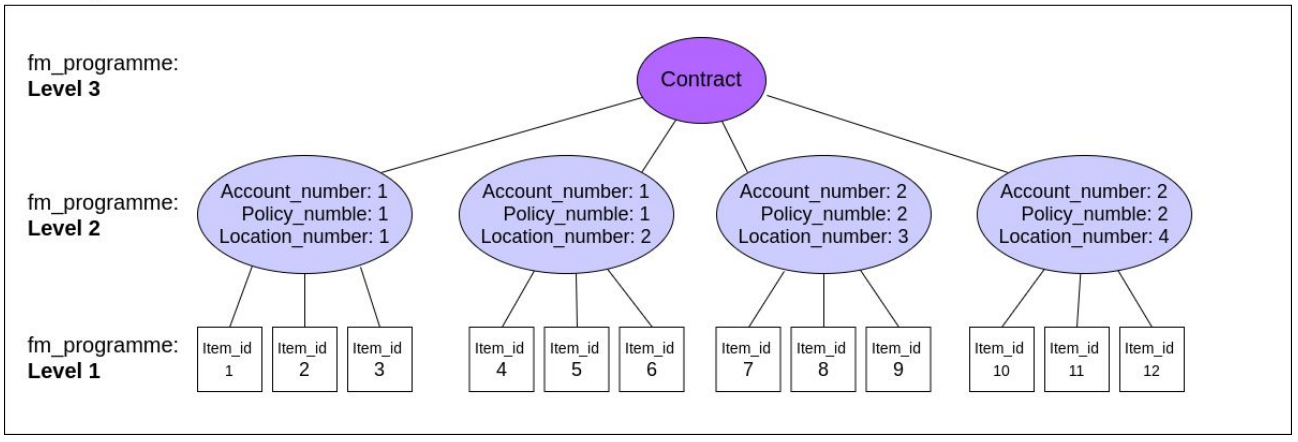
ri_scope.csv

ReinsNumber	PortfolioNumber	AccountNumber	PolicyNumber	LocationNumber	CededPercent	RiskLevel
1	1	1	1	1	0.1	LOC
1	1	1	1	2	0.1	LOC
1	1	2	2	3	0.1	LOC
1	1	2	2	4	0.1	LOC
2	1	1	1	1	0.2	LOC
2	1	1	1	2	0.2	LOC
2	1	2	2	3	0.2	LOC
2	1	2	2	4	0.2	LOC

Layer 1



Layer 2



fm_policytc.csv

layer_id	level_id	agg_id	profile_id
1	2	1	7
1	1	1	3
1	1	2	4
1	1	3	5
1	1	4	6
2	2	1	12
2	1	1	8
2	1	2	9
2	1	3	10
2	1	4	11

fm_profile.csv

profile_id	calcrule_id	deductible1	deductible2	deductible3	attachment	limit	share1	share2	share3
1	14	0	0	0	0	0	0	0	0
2	12	0	0	0	0	0	0	0	0
3	24	0	0	0	0	9999999999999	0.1	1	1
4	24	0	0	0	0	9999999999999	0.1	1	1
5	24	0	0	0	0	9999999999999	0.1	1	1
6	24	0	0	0	0	9999999999999	0.1	1	1
7	23	0	0	0	0	9999999999999	0	1	1
8	24	0	0	0	0	9999999999999	0.2	1	1
9	24	0	0	0	0	9999999999999	0.2	1	1
10	24	0	0	0	0	9999999999999	0.2	1	1
11	24	0	0	0	0	9999999999999	0.2	1	1
12	23	0	0	0	0	9999999999999	0	1	1

fm_programme.csv

from_agg_id	level_id	to_agg_id
1	2	1
2	2	1
3	2	1
4	2	1
1	1	1
2	1	1
3	1	1
4	1	2
5	1	2
6	1	2
7	1	3
8	1	3
9	1	3
10	1	4
11	1	4
12	1	4