D+ Monitoring
Full Family Tree Monitoring Product

Onboarding Document

External Document

Document Author	: Kevin O'Connor
Dev Manager	: John McGoldrick
Published Date	: 10th May 2019
Version Number	: 1

Revision History & Sign-off Sheet

Change Record

Version	Date	Name	Revision Description
V1	10th May 2019	Kevin O'Connor	This is for the early adopter architects to get up to speed on how to set up a registration. D&B Documentation (Swagger) will be updated subsequently.

Reviewers

Name	Version Approved	Role	Date
John McGoldrick	V1	Development Manager – D+ Monitoring	10th May 2019

Distribution

Name	Date
Marty Walls	10th May 2019
Ayan Basu	10th May 2019

Table of Contents

1.0 Introduction

The purpose of this document is to provide an overview of the D+ Monitoring Family Tree Product. The audience will be the integration architects of the early adopter users.

This document will outline how to set up a registration and receive the Family Tree Monitoring artifacts. Once the first architects have created registrations and received artifacts, regular communication and assitance will be provided by the D&B TAM's – to ensure smooth onboarding and feedback to D&B Technology.

After review of this document by the Product Owner and wider Product Team, the external D&B Documentation site (swagger) will be updated accordingly.

The UAT testcases were compiled by the Product Owner using inputs and learnings taken in D&B Product Workshops hosted in 2017 & 2018.

1.1 Project Overview

The supported product is a newly created one called:

- Full_Family_Tree

1.2 Scope

1.2.1 In Scope

- Internal movement of a DUNS in a single tree (e.g. a DUNS changes its Immediate Parent but still has the same GU) – will register as "moved" in this notification file. "moved" is also used for movement of DUNS to another tree.
- Creation of Family Tree Registrations using the Internal Dashboard.

1.2.2 Out of Scope

- Stop Distribution or Transferred DUNS are not in scope for the SUMMARY file. They are in scope during the registration and Add DUNS steps (as per existing monitoring registrations.)
- The array of "deleted" has been deemed out of scope as it will be covered by other scenarios in the tree.
- Filtering of elements (jsonPathInclusion or jsonPathExclusion).
- DAILY, WEEKLY, MONTHLY frequencies.
- Universe is not required for Family Tree Registrations.
- A seed file on demand is not currently available.

2.0 Architecture

2.1 Architectural Approach

Design and architural considerations were taken at various stages over the past 2 years. To leverage the benefits of the existing Monitoring architecture, code and functionality – the aspects which could remain the same for the end-user were maintained. For example – the registration process will be a familiar flow, and most of the API calls can be re-used.

Due to the uniqueness of some of the requirements compared to existing monitoring Products, it was clear that a different approach on some facets was required (see below).

2.1 Deviation from the existing Monitoring Functionality

The following are specific to Family Tree Monitoring Registrations:

- The only notification type that is permitted is SUMMARY. This is a new notification artefact which differs in format to any of our existing notifications. SUMMARY can only be used on Family Tree Registrations.
- SEED on demand is not currently available. A seed file will be provided in the following scenarios:
    1. A new registration is created.
    2. An edit registration is initiated.
    3. A new GU is created when a DUNS of Interest moves to a tree that is not already being monitored on.
- D+ Monitoring Family Tree is INTRA_DAY only as the SUMMARY notification acts as linear picture of changes – i.e. aggregation using other frequencies would not provide insight or value.
- Direct+ will provide the full Family Tree artefact and perform error and exception handling. D+ Monitoring's primary responsibility is user registration management and routing of artefacts.
- Family Tree Registrations will be set to Unsuppressed by default, upon creation. This is to ensure the sequential flow of the artefacts is maintained (i.e. the SUMMARY notifications will build out from the initial SEED)

3.0 Registration

The Family Tree Monitoring registration has replicated how the existing monitoring registrations look for the end-user – with some slight deviations outlined below.

3.1 Parameters specific to Family Tree

Users should be aware of the following unique attributes for a Family Tree Monitoring Registration:

- productId is FULL_FAMILY_TREE
- seedData is TRUE
- notificationType is SUMMARY

3.2 Sample Create Registration

Request Param	**FT**	**Example**
reference	Y; Required	FT_Test1
description	Y; Optional	Test Family Tree Registration
fileTransferProtocol	Y; Required • S3 or FTP path	arn:aws:s3:::monitoring/ExampleFolder/${timestamp }
destinationType	Y; Required • S3 or FTP	S3
deliveryFrequency	Y; Required • INTRA_DAY	INTRA_DAY
productID	Y; Required • FULL_FAMILY_TREE	FULL_FAMILY_TREE
versionID	Y; Required • v1	v1
duns	Y; Required	(upload list of DUNS as per LOD registration)
seedData	Y; Required (defaults to TRUE) • TRUE	TRUE
notificationType	Y; Required • SUMMARY	SUMMARY
email	Y; Required	test@test.com
encryptionKey	Y; Conditionally Required for S3	arn:aws:s3:::customer-bucket-name/*

4.0 File Artifacts

The following artifacts are delivered to the users:

4.1 DUNSMap (Mapping file)

- Given that upon successful creation of a create or edit Registration, when the DUNSMap file is created, then it will contain the latest mapping of all of the duns to their corresponding global ultimate duns:

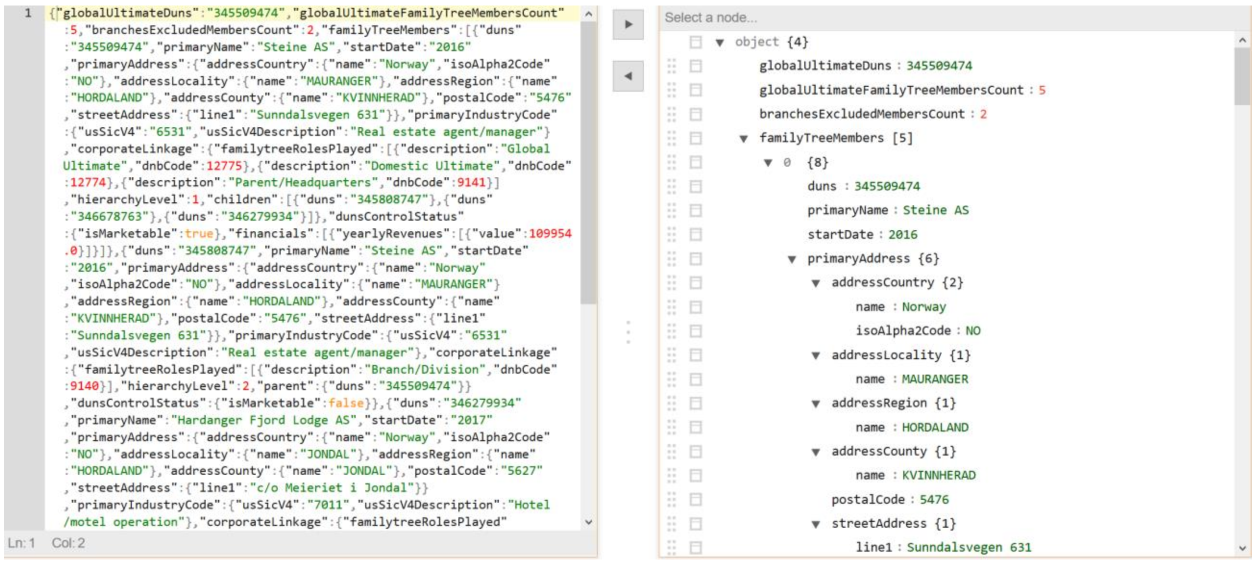
  {"duns":"804735132","gu":"804735132"}

- If a DUNS of Interest submitted is a standalone DUNS (i.e. no GU) then the following response will be returned:

  {"duns":"804735132","gu":"standalone"}

4.2 Seed

- Given that some existing monitoring users are used to the Family Tree API, we have replicated the fabrication of this file to match this. It contains latest family trees available:



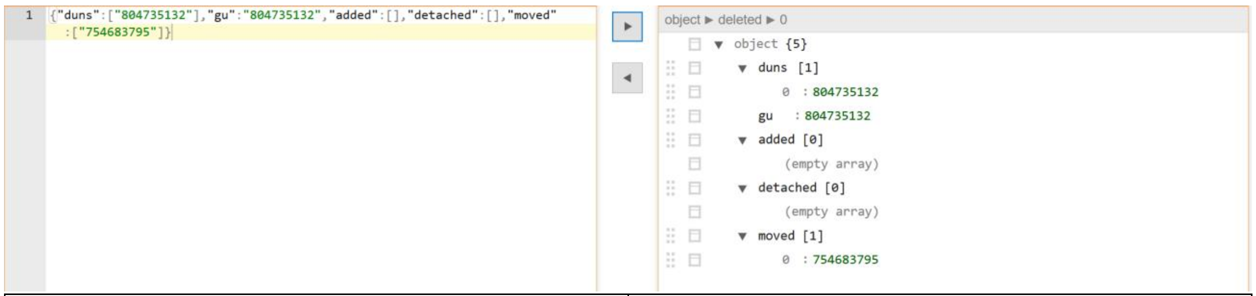
```
{"globalUltimateDuns":"345509474","globalUltimateFamilyTreeMembersCount"
:5,"branchesExcludedMembersCount":2,"familyTreeMembers":[{"duns"
:"345509474","primaryName":"Steine AS","startDate":"2016"
,"primaryAddress":{"addressCountry":{"name":"Norway","isoAlpha2Code"
:"NO"},"addressLocality":{"name":"MAURANGER"},"addressRegion":{"name"
:"HORDALAND"},"addressCounty":{"name":"KVINNHERAD"},"postalCode":"5476"
,"streetAddress":{"line1":"Sunndalsvegen 631"}},"primaryIndustryCode"
:{"usSicV4":"6531","usSicV4Description":"Real estate agent/manager"}
,"corporateLinkage":{"familytreeRolesPlayed":[{"description":"Global
Ultimate","dnbCode":12775},{"description":"Domestic Ultimate","dnbCode"
:12774},{"description":"Parent/Headquarters","dnbCode":9141}]
,"hierarchyLevel":1,"children":[{"duns":"345808747"},{"duns"
:"346678763"},{"duns":"346279934"}]},"dunsControlStatus"
:{"isMarketable":true},"financials":[{"yearlyRevenues":[{"value":109954
.0}]}]},{"duns":"345808747","primaryName":"Steine AS","startDate"
:"2016","primaryAddress":{"addressCountry":{"name":"Norway"
,"isoAlpha2Code":"NO"},"addressLocality":{"name":"MAURANGER"}
,"addressRegion":{"name":"HORDALAND"},"addressCounty":{"name"
:"KVINNHERAD"},"postalCode":"5476","streetAddress":{"line1"
:"Sunndalsvegen 631"}},"primaryIndustryCode":{"usSicV4":"6531"
,"usSicV4Description":"Real estate agent/manager"},"corporateLinkage"
:{"familytreeRolesPlayed":[{"description":"Branch/Division","dnbCode"
:9140}],"hierarchyLevel":2,"parent":{"duns":"345509474"}}
,"dunsControlStatus":{"isMarketable":false}},{"duns":"346279934"
,"primaryName":"Hardanger Fjord Lodge AS","startDate":"2017"
,"primaryAddress":{"addressCountry":{"name":"Norway","isoAlpha2Code"
:"NO"},"addressLocality":{"name":"JONDAL"},"addressRegion":{"name"
:"HORDALAND"},"addressCounty":{"name":"JONDAL"},"postalCode":"5627"
,"streetAddress":{"line1":"c/o Meieriet i Jondal"}}
,"primaryIndustryCode":{"usSicV4":"7011","usSicV4Description":"Hotel
/motel operation"},"corporateLinkage":{"familytreeRolesPlayed"
Ln: 1   Col: 2
```

```
Select a node...
▼ object {4}
    globalUltimateDuns : 345509474
    globalUltimateFamilyTreeMembersCount : 5
    branchesExcludedMembersCount : 2
  ▼ familyTreeMembers [5]
    ▼ 0  {8}
        duns : 345509474
        primaryName : Steine AS
        startDate : 2016
      ▼ primaryAddress {6}
        ▼ addressCountry {2}
            name : Norway
            isoAlpha2Code : NO
        ▼ addressLocality {1}
            name : MAURANGER
        ▼ addressRegion {1}
            name : HORDALAND
        ▼ addressCounty {1}
            name : KVINNHERAD
          postalCode : 5476
        ▼ streetAddress {1}
            line1 : Sunndalsvegen 631
```

4.3 Summary

- For each GU we show the changes that have occurred since the last cutoff. They are grouped by change type and include the list of 'DUNS of Interest' first:

{"duns":["804735132"],"gu":"804735132","added":[],"detached":[],"moved":["754683795"]}



```
{"duns":["804735132"],"gu":"804735132","added":[],"detached":[],"moved"
:["754683795"]}
```

```
▼ object {5}
    ▼ duns [1]
          0 : 804735132
      gu  : 804735132
    ▼ added [0]
          (empty array)
    ▼ detached [0]
          (empty array)
    ▼ moved [1]
          0 : 754683795
```

States	**High-Level Definition**
Added	DUNS added to tree.
Detached	DUNS moved to become a standalone DUNS. Has no parents or children.
Moved	DUNS moved from one GU to another, it now has a new GU. *Or* DUNS has moved from under one parent to under another (whilst remaining under the same GU).

- If there is no DUNS in the array then the array name will appear with brackets [] e.g. "detached":[] . This maintains the structure of the summary information for each GU.

4.4 Exceptions File & Header File

As per existing monitoring format.

5.0 Exisiting Monitoring API Calls

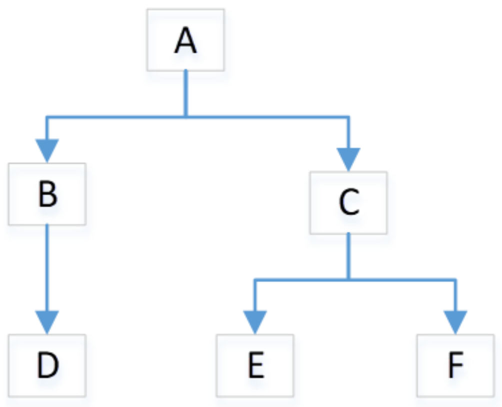
The existing API calls listed will work for Family tree Monitoring also:

- Get Details
- Get Status
- List Pending Deliveries
- DUNS check
- DUNS export
- Push notifications
- Suppress
- Unsuppress
- Delete registration
- GU Count (Note – this will not be updated during the “edit” DUNS action so may be out of sync until the next publication/distribution run. As INTRA_DAY is the frequency cadence then we think this has a minor impact. It will be updated during “create” and every run.)

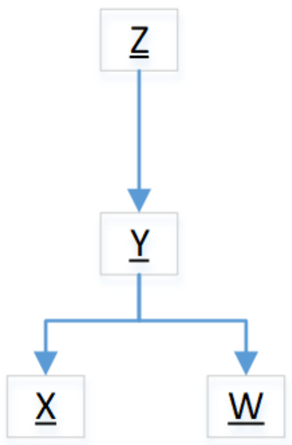
9.0 UAT Test Cases

9.1 Starter Trees

Tree A:



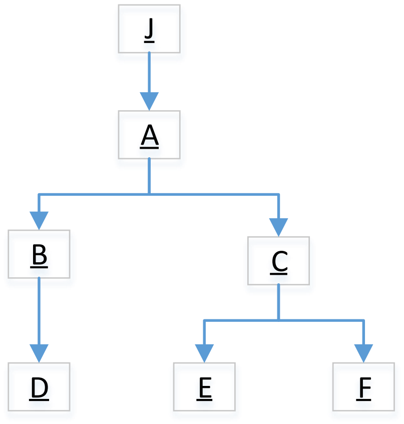
Tree Z:

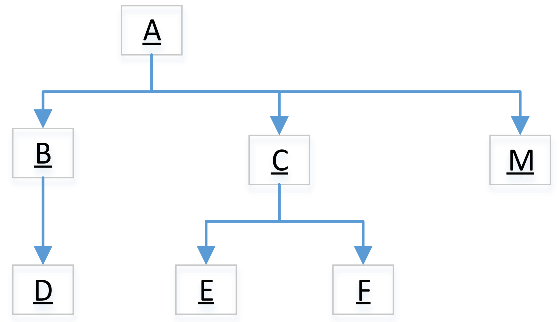
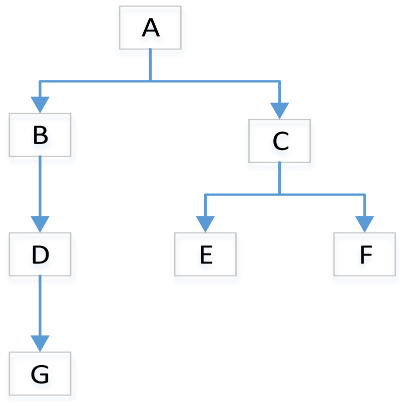


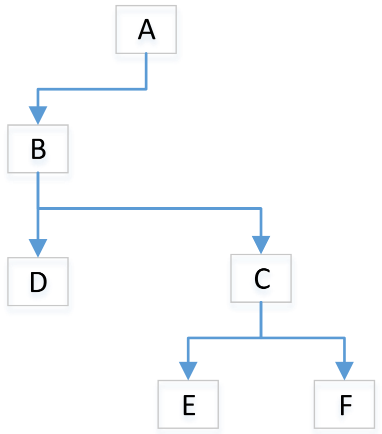
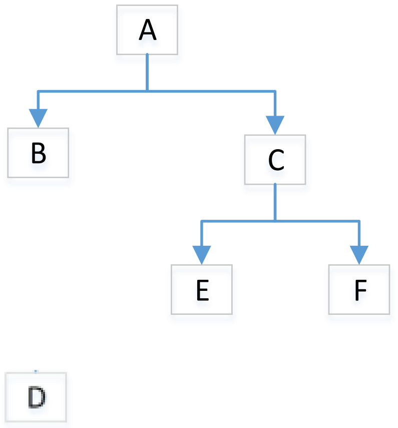
A is the GU

Z is the GU

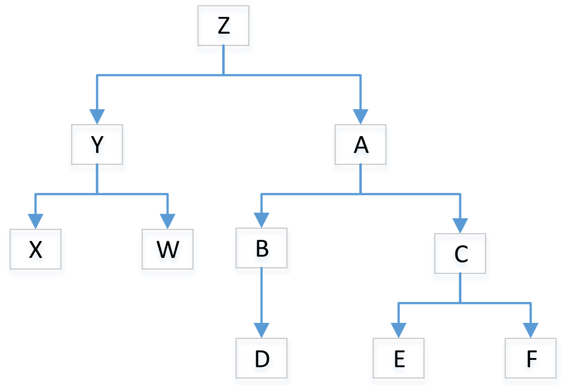
Scenario ID:	**Change:**	**Sample:**	**Notification File Format:**

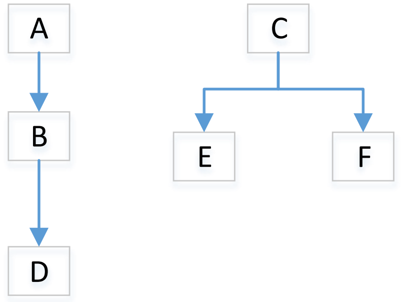
Scenario 1	A gets new parent; becomes new child under J. **J is an existing entity.**		**Format**
			`{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "A", "added":[], "detached":[], "moved":["A", "B", "C", "D", "E", "F"]}`
			`{"Duns of Interest":[],"gu": "J", "added":["A", "B", "C", "D", "E", "F"], "detached":[], "moved":[]}`
			
			Logic:
			Original Tree A-F has been moved under a new parent of J (“moved”). Tree A-F is “added” to the existing entity J.
			
			In this scenario D+ Monitoring will have logic to send out a seed file, because we are guaranteed that one of the DUNS from A to F must be DUNS of interest so will then be registered under a new GU of J – this will trigger a seed.

Scenario ID:	Change:	Sample:	Notification File Format:
Scenario 2	A gets new child		**Format** `{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "A", "added":["M"], "detached":[], "moved":[]}` Logic: M is added to this tree.
Scenario 3	D gets new child		**Format** `{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "A", "added":["G"], "detached":[], "moved":[]}` Logic: G is added to this tree.

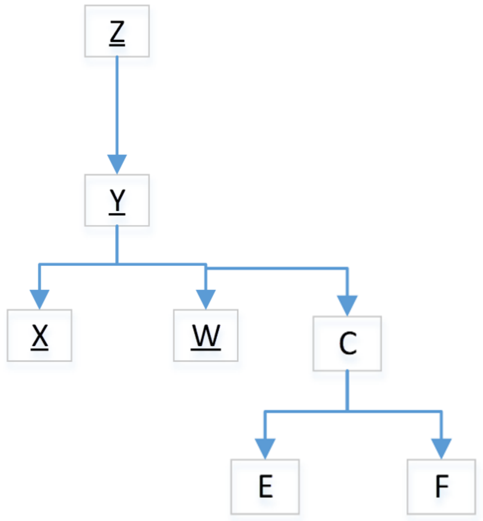
Scenario ID:	Change:	Sample:	Notification File Format:
Scenario 4	C moves under B		**Format** `{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "A", "added":[], "detached":[], "moved":["C"]}` Logic: This move of a child from one parent to another (under the same GU) is recorded as a "moved". *In this scenario there is a "moved" notification, but no "added" notification. This is because the GU has not changed for this DUNS in question. Therefore, the end user can derive that this DUNS in question has moved to a different parent under the same GU.*
Scenario 5	D leaves tree; changes to standalone		**Format** `{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "A", "added":[], "detached":["D"], "moved":[]}` Logic: D becomes a standalone DUNS (orphan with no parents or children). As it does not join another tree, there are no other notifications.

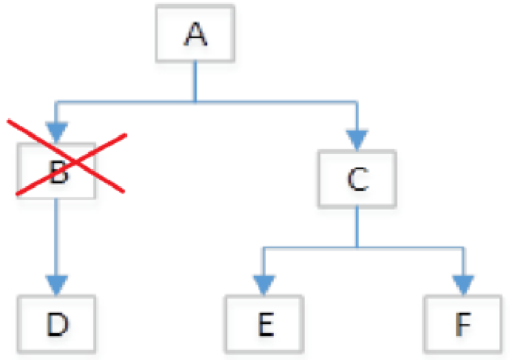
Scenario ID:	Change:	Sample:	Notification File Format:

Scenario 6	A moves to another existing tree		**Format**
			
			`{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "A", "added":[], "detached":[], "moved":["A", "B", "C", "D", "E", "F"]}`
			`{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "Z", "added":["A", "B", "C", "D", "E", "F"], "detached":[], "moved":[]}`
			
			Logic:
			All DUNS including the GU itself are moved from A and added to Z.

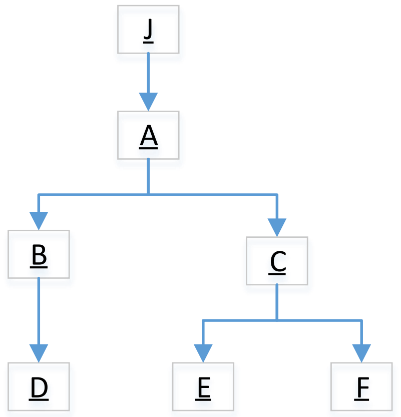
Scenario 7	C moves to its own tree		**Format**
			
			`{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "A", "added":[], "detached":[], "moved":["C", "E", "F"]}`
			`{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "C", "added":["E", "F"], "detached":[], "moved":[]}`
			
			Logic:
			C, E and F have left Tree A ("moved"). C is now a new GU with E and F as children DUNS ("added").
			
			In this scenario D+ Monitoring will have logic to send out a seed file, if any of C, E or F are DUNS of interest.

Scenario ID:	Change:	Sample:	Notification File Format:

Scenario 8	**Scenario 8:**
	
	C, E and F move from Tree A to under a child in Tree Z
			**Format**
			
			`{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "A", "added":[], "detached":[], "moved":["C", "E", "F"]}`
			`{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "Z", "added":["C", "E", "F"], "detached":[], "moved":[]}`
			
			Logic:
			All DUNS including the GU itself are moved from A and added to Z.

~~**Scenario 9**~~	~~The DUNS B is deleted in D&B~~		**Out of scope:** Deemed out of scope due to the following reasons: *A that a delete scenario is the same as any other delink scenario where a DUNS is removed from a tree. The fact that it came from a delete isn't relevant for informing of the delink having happened. The delete in this case would be the reason that the company became delinked from the tree and the reason is not something that we vend today and if we were going to communicate that a delete caused a delink I would expect it would be only when we are in a position to communicate the reason for all delink scenarios (merger, divestment, Out of Business etc etc) and Delete would be one of many possible values in the reason text.*

Scenario ID:	Change:	Sample:	Notification File Format:

Scenario 10	A gets new parent; becomes new child under J. **J is a new entity.**		**Format**
			`{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "A", "added":[], "detached":[], "moved":["A", "B", "C", "D", "E", "F"]}`
			`{"Duns of Interest":[`*`Uploaded DUNS of Interest for this tree will be here`*`],"gu": "J", "added":["A", "B", "C", "D", "E", "F"], "detached":[], "moved":[]}`
			
			Logic:
			Original Tree A-F has been moved under a new parent of J (“moved”). Tree A-F is “added” to entity J.
			
			If new GU is created and the GU or children are D+ Monitoring DUNS of interest then we will send a seed.