MTD API Refactoring to Enable New Entity Types: Estimate

22 September 2020

Contributors

- Toby Porter, MTD Front Facing API Architect
- Mark Kelly, MTD Front Facing API Technical Lead

1. Problem Statement

The current MTD ITSA individual APIs are not suitable to reuse for other entity types because they are tied to use of a nino. This makes delivery of additional return types expensive and time consuming. We want to investigate the possibility of making a breaking change to the individual APIs to enable them to be reused for Trusts/Partnerships etc.

1.1 High Level Assumptions

1. There are [initially] 3 additional entities that need to be supported: Partnerships, Trusts & Pension Schemes
2. Return requirements are broadly the same as Individuals but may differ by 10%, 30% & 50% respectively i.e. Individuals is considered to be a super-set of Partnerships etc.
3. More entities may need to be supported in future
4. Calculations API would need minimal modifications

1.2 Definitions

- Sprint - One [development] sprint is two weeks in length

2. Approach

The strategic approach is to re-register all the existing *individuals* endpoints under entity agnostic contexts.

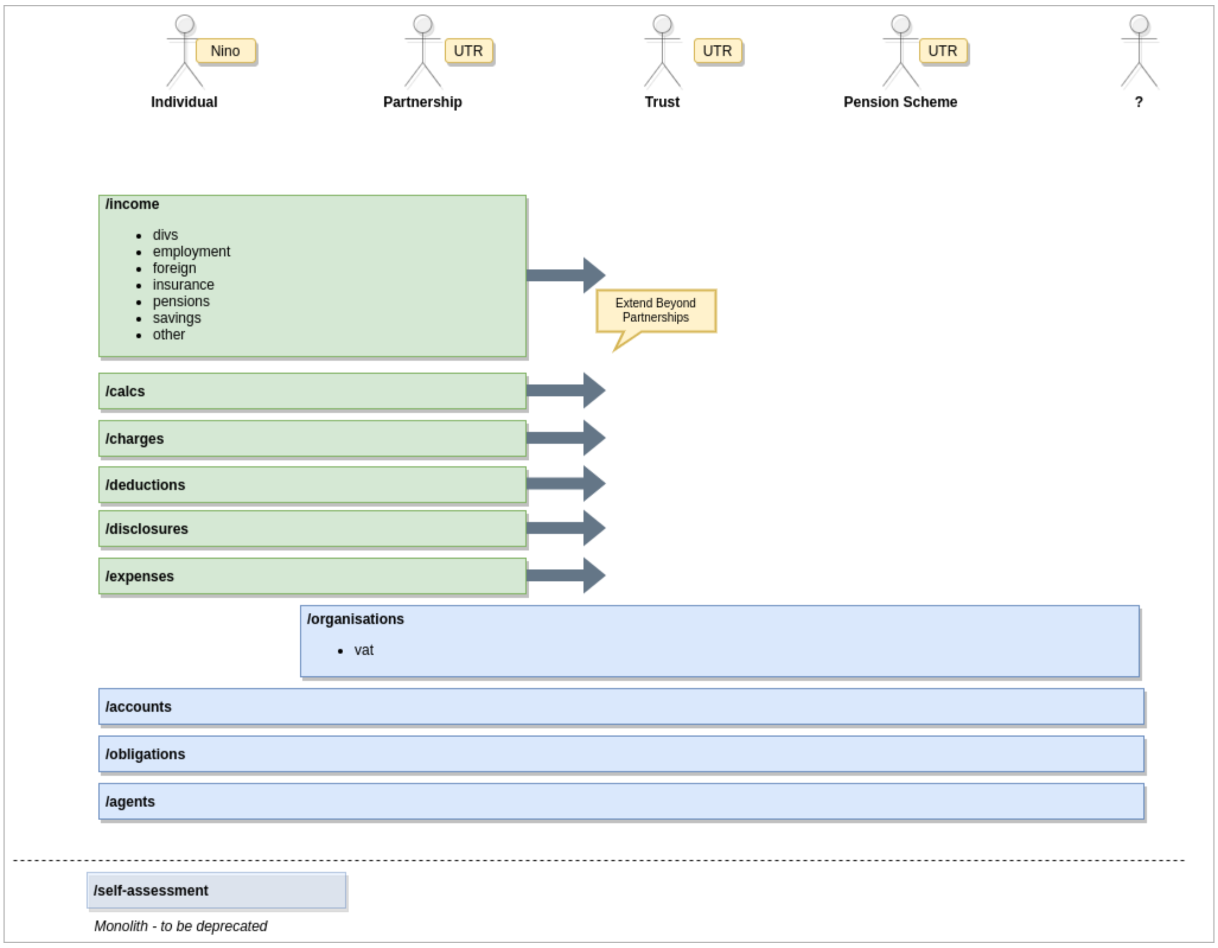

Figure 1: MTD API To Be Contexts

2.1 High Level Scope

2.2 APIs in Development

- Refactor to entity agnostic top-level context(s)
- Code
- HATEOS links
- Tests

Est: **1 Sprint**

2.3 APIs in Sandbox

- Refactor to entity agnostic top-level context(s)
- Code
- Unpublish services & republish under new context
- HATEOS links
- Documentation
- Tests
- Re-publish to API platform
- Satisfy Transaction Monitoring requirements
- Communicate changes to third party vendors

Est: **6 Sprints**

2.3 APIs in Production

- Refactor to entity agnostic top-level context(s)
- Create services for new contexts
- Code
- HATEOS links
- Documentation
- Tests
- Re-publish to API platform
- Satisfy Transaction Monitoring requirements
- Communicate changes to third party vendors

Est: **8 Sprints**

2.3 Deprecate “Monolith API”

- Refactor to entity agnostic top-level context(s)
- Code
- HATEOS links
- Documentation
- Tests
- Re-publish to API platform
- Satisfy Transaction Monitoring requirements
- Communicate changes to third party vendors

Est: **TBC**

3. Risks & Issues

3.1 Risks

#	Risk	Probability	Impact	Mitigation/Contingency
1.	Increased complexity and chance of creating new “monolithic” APIs if entity logic diverges	Med	High	Monitor and split if necessary…
2.	API Platform reject the request for additional top level contexts	High	High	Start engagement with platform team and get backing of MTD programme if needed. (MTD APIs are the largest “customer”)
3.	Pushback from third party vendors who have developed against existing *individuals* endpoints (Expectations of 3PV for Partnerships)	Medium	Medium	Follow standard breaking change process of keeping deprecated endpoints available for 6mths
4.	Entity agnostic URI paths potentially make it harder for third party vendors to explore and identify the correct API for their needs	Medium	Medium	More thorough guidance, more flexible grouping options on the API Platform (tagging?).

3.2 Issues

#	Issue	Resolver	Resolution
1.	The API Platform require a level of specificity in the naming so that top-level contexts can be shared by other teams.	MTD API Team & API Platform	
2.	How should the front facing APIs determine the type of entity?	CDIO	
3.	What will be the impact on downstream systems DES, ITSD, ETMP etc	CDIO	

#	Issue	Resolver	Resolution
4.	API authentication is currently based on enrolment ID of the individual which is mapped to the Nino. Discovery work needed to understand the impact on this process.	CDIO/MTD API Team	