1 Operations, Primitives and Algorithms

The following sections introduce, define and explain Operations, Primitives and Algorithms generally using the Terminology presented below. Operations are the building blocks of Primitives whereas Primitives are the building blocks of Algorithms. The definitions which follow are flexible enough to support implementation across programing languages but have been inspired by the core concepts found within Lisp. The focus of these sections is to define the properties of and interactions between Operations, Primitives and Algorithms in a general way which doesn't place unnecessary bounds on their range of possible functionality with respect to processing xAPI data.

1.1 Terminology

In the subsections and sections which follow, (s) indicates one or more

1.1.1 Scalar

Singular value x of a fundamental JSON type as described by JSON Schema

1.1.2 Collection

a n-tuple of items x such that

$$X = < x_i..x_n..x_j >$$

where

$$i \leq n \leq j \Rightarrow i \prec n \prec j \iff i \neq n \neq j$$

1.1.3 Key

A lookup k for a v within a kv where $k = x \lor X$

1.1.4 Value

a piece of data v where $v = x \lor X$

1.1.5 Key Value pair

Association between a k and v where

$$k \mapsto v$$

such that

$$kv = k \mapsto v$$

and a collection of Key Value pair(s) is defined as

$$KV = < kv_i..kv_n..kv_j >$$

such that

$$k_n \mapsto v_n$$

and all k within KV are unique

$$i_k \neq n_k \neq j_k$$

but the same is not true for all v within KV

$$i_v = n_v \vee i_v \neq n_v \;\; i_v = j_v \vee i_v \neq j_v \;\; j_v = n_v \vee j_v \neq n_v$$

1.1.6 Statement

Immutable collection of Key Value Pair(s) conforming to the xAPI Specification

1.1.7 Algorithm State

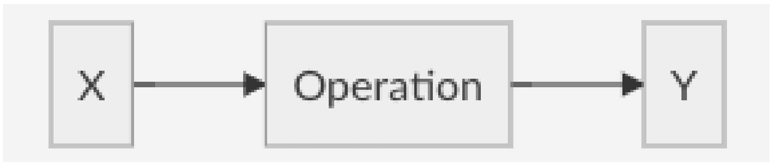
Mutable collection of Key Value Pair(s)

1.1.8 Option

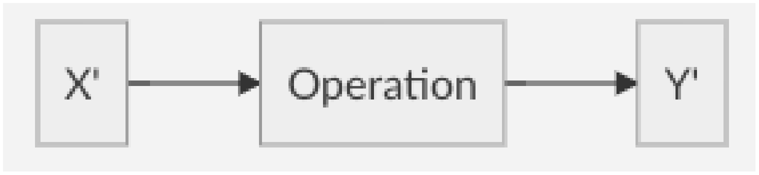
Collection of Key Value Pair(s) which alter the result of an Algorithm

1.2 Operation

Given an input X, an Operation produces output Y



If X changes to X' then the Operation results in Y' instead of Y



1.2.1 Domain

Any of the following

- Key(s)
- Value(s)

- Key Value pair(s)
- Statement(s)
- Algorithm State

1.2.2 Range

Any of the following dependent upon the Domain and Functionality of the Operation

- Key(s)
- Value(s)
- Key Value pair(s)
- Statement(s)
- Algorithm State

1.2.3 Formal Definition

A relationship between input and output data which will result in the same Y given the same X

$$Operation(X) = Y \wedge \ Operation(X') = Y'$$
$$\Rightarrow$$
$$Y = Y' \iff X = X' \wedge Y \neq Y' \iff X \neq X'$$

When X is a collection, that collection consists of one or more members x_n within the range $i..j$ such that

$$i \leq n \leq j \Rightarrow i \prec n \prec j \iff i \neq n \neq j$$

which allows collection X to be defined as

$$X =< x_i..x_n..x_j >$$

and the output of $Operation(X)$ to be defined as

$$Y =< y_i..y_n..y_j >$$

such that the mapping from $Operation(X) \rightarrow Y$ can be defined as

$$Operation(x_i) \mapsto y_i \wedge Operation(x_n) \mapsto y_n \wedge Operation(x_j) \mapsto y_j$$

which implies that the collection Y has the same ordering as collection X

$$i_{Operation(x)} = i_y \ \wedge \ n_{Operation(x)} = n_y \ \wedge \ j_{Operation(x)} = j_y$$

and allows us to define the *input* $\rightarrow$ *output* relationship of an *Operation* in terms of collection members

$$Operation(x) = y$$

which establishes how an *Operation* responds to non-distinct values within a Collection

$$Operation(x_n) = y_n$$

$$Operation(x_{n'}) = y_{n'}$$

$$Operation(x_{n'+1}) = y_{n'}$$

$$\iff$$

$$x_{n'} \equiv x_{n'+1} \wedge x_{n'} \not\equiv x_n$$

$$\Rightarrow$$

$$x_n \not\equiv x_{n'+1}$$

$$\Rightarrow$$

$$Operation(x_{n'}) = Operation(x_{n'+1}) \neq Operation(x_n)$$