# Data Standards Body
## Technical Working Group
## Decision 013 – Primitive Data Types

*Contact: James Bligh*

*Publish Date: 23rd September 2018*

*Decision Approved By Chairman: 4th October 2018*

## Context

To ensure consistent implementation and to increase understanding and consumption the payloads of the API end points should be strongly typed.  It should be clear what data is expected in every field in the API end points.

To facilitate this a series of primitive data types that will be commonly used should be defined. Fields in the JSON payloads will not be constrained to use these types but they should be used by preference where possible.

## Decision To Be Made

Determine the initial set of primitive data types to be used for defining fields in JSON payloads.

## Feedback Provided

The original proposal and the associated feedback can be found at:
https://github.com/ConsumerDataStandardsAustralia/open-banking/issues/13

Feedback on this proposal was supportive of the original proposal with some minor additions and amendments that have been accommodated in the final decision below.

## Decision For Approval

The primitive data types to be supported are described below.

| Type | Description | Valid Examples |
|---|---|---|
| String | Standard UTF-8 string but unrestricted in content. Any valid Unicode character can be used. | |
| ASCIIString | Standard UTF-8 string but limited to the ASCII character set. | |
| Boolean | Standard JSON boolean | true<br>false |
| Enum | String representing an option from a defined list of value<br>• All possible values should be provided<br>• Values should be in all caps<br>• Spaces should be replaced with under bars '_'<br>Values should be limited to the ASCII character set | "OPTION1"<br>"ANOTHER_OPTION"<br>"VAL_ABC_123" |
| PositiveInteger | A positive integer inclusive of zero | 0<br>1<br>10000 |
| NegativeInteger | A negative integer inclusive of zero | 0<br>-1<br>-10000 |
| Integer | Any positive or negative integer inclusive of zero | 1<br>0<br>-1 |
| Number | A standard floating point number. Can be positive, negative or zero | 0.1<br>-100.09<br>10<br>90.09 |
| DateTimeString | Combined Date and Time string as per RFC-3339 (labelled *date-time* in the RFC). UTC time should always be used | "2007-05-01T15:43:00.12345Z"<br>"2012-12-25T15:43:00-08:00"<br>"1997-01-12T15:43:00.121Z" |

| | | |
|---|---|---|
| DateString | Date string as per RFC-3339 (labelled *full-date* in the RFC). UTC time should always be used | “2007-05-01”<br>“2012-12-25” |
| TimeString | Time string as per RFC-3339 (labelled *full-time* in the RFC). UTC time should always be used | “15:43:00.12345Z”<br>“15:43:00-12:00” |
| CurrencyString | Standard 3 character currency codes as per ISO-4217 | “AUD”<br>“USD”<br>“GBP” |
| RateString | A string representing a percentage interest rate<br>• A positive number (or zero)<br>• At least 1 and up to a total of 16 significant digits before decimal point<br>• Up to 16 digits following the decimal point<br>• No formatting, eg thousand separating commas | “82”<br>“0.05”<br>“12.3456789”<br>“99.123456789123” |
| AmountString | A string representing an amount of currency.<br>• A positive, zero or negative number<br>• Negative numbers identified with a ‘-‘<br>• No currency symbols should be supplied<br>• At least 1 and up to a total of 16 significant digits before decimal point<br>• Minimum 2 digits following a decimal point (more digits allowable but only if required)<br>• No additional formatting, eg thousand separating commas | “0.01”<br>“10.00”<br>“1234567.89”<br>“-1001.23”<br>“1.999” |
| MaskedPANString | Masked credit card number. Lower case ‘x’ should be used to mask numbers and only the last four digits should be exposed to facilitate identification. | xxxxxxxxxxxx1234 |