SData 2.0 Core

Version 1.0

Motivation and purpose

A clear and concise definition of SData is one of the primary goals for the 2.0 version.

The SData standard is a comprehensive specification capable of handling even the most complex business cases that might require synchronization, paging, discoverability, etc. In the past, teams building SData providers believed they had to implement everything mentioned in the SData 1.x standard, even features not required by their particular business case. In some instances this lead to the perception that building an SData provider was more difficult to create than it was really the case.

SData 2.0 can be viewed as a web toolkit consisting of a small set of required core features and many optional features that solve frequently encountered problems. This document will capture and define the small number of core features that make up the 'SData 2.0 core'.

In addition to clarifying and simplifying the standard, this document also introduces JSON as a new optional format for SData payloads. JSON is ideally suited to some business cases like mobility because it's less verbose then ATOM/XML. This document will not provide a complete JSON specification but it will compare and contrast JSON and ATOM formats defined in SData 2.0.

Aims

This paper designates and describes the central aspects of SData, yielding the 'core' of the specification. The document does not contain the official text of the core itself - it points out the fundamentals, leaving the global community presentation of the topics open for future work.

This document will:

- Identify the SData 2.0 fundamental aspects and establish the relation to their counterparts in the 1.x version of the standard
- Re-define compliance levels(MUST/SHOULD/MAY) on core aspects
- Deprecate outdated features or solutions

Fundamental SData aspects

SData 2.0 ensures that version 1.x compliant implementations are equally compliant with the new version of the standard. This is achieved by:

- Relaxing existing compliance levels thus making even more aspect optional
- Adding new, optional features
- Deprecation of features where better technological answers have emerged in the meantime

The SData core identifies the aspects essential to the philosophy of the standard. An essential aspect does not imply that its components must be implemented in their entirety to achieve compliance; the degree of choice is indicated by the (MUST/SHOULD/MAY) qualifiers; these may be tightened further by underlying contracts.

The SData fundamental aspects are recognized to be:

- SData URL
- SData payload formatting
- Requesting content formats
- Operations
- Authorization
- Status and error conditions

These are discussed in turn in the following chapters of this document. Unless specifically stated, the definitions in the SData 1.x standard maintain their validity in the SData 2.0 standard.

SData URL

The SData URL is responsible for resource addressing and represents the tie-in with the REST architectural style. The current specification (Chapter 2: Anatomy of an SData URL) is fairly restrictive due to the initial, more normative approach adopted at the time.

The proposals presented in the upcoming sub-sections remove a number of current rules and restrictions. The intended effect is increased simplicity coupled with lending applications the ability to construct URLs that fit better with their natural structure and underlying technologies.

Liberalization of the URL structure up to the query component

In SData 2.0 the URL structure prior to the Query component is freely definable by the provider. This applies specifically to the existence and structure of the following URI Path[1]sub-components as defined by SData in the Resource Collection URL section:

- Virtual directory:
  - MAY be '`sdata`', which is preferred
  - MAY be a set of several sub-segments
- Contract name:
  - MAY be omitted if it is the only contract to be exposed by the provider
  - MAY comprise several sub-segments. A meaningful sub-segment is the contract version that an application MAY expose
- Application name:
  - MAY be provided. It is recommended that the application name be a part of the URL.
- Dataset:
  - MAY be omitted if only one dataset is supported by the provider
  - MAY contain several sub-segments reflecting a hierarchical dataset selection structure
  - The composition formalism described in 1.x standard remains valid

The requirements imposed by version 1.x (Section 2.1 Resource Collection URL) become guidelines[2] in SData 2.0, meaning that implementations are encouraged (but not forced) to act accordingly.

[1] URI Path as defined in section 3 of RFC 3986

Reserved segments at the application level

In SData 2.0, none of the segments mentioned below are required – these MAY be present if required by the underlying contract:

- `$schema:` amendment to the 1.x version that required the presence of the segment
- `$service:` clarification to the 1.x version
- `$queries:` clarification to the 1.x version
- `$batch:` amendment to the 1.x version where the `$batch` segment was tied exclusively to a resource kind URL

The corresponding SData 1.x definitions are found under:

- 2.4 Service Operation URL
- 2.5 Named Query URL
- 2.6 Template resource URL
- 2.7 Resource schema URL
- 2.9 Intermediate URLs
- 13.1 Batch URL

Schema requirement

In SData 2.0 the presence of the schema is optional. This applies equally to application-wide and resource kind schemas.

SData 1.x required a schema to be present (see 2.7 Resource Schema URL). While it is recognized that schemas are a meaningful component in application-to-application scenarios, providing and maintaining a schema in most other scenarios is difficult and costly. SData 2.0 lifts the existing requirement and leaves it up to the contract to require the presence of a schema.

SData payload elements

The structure SData content is aligned with the ATOM specification, according to the initial direction we took. ATOM divides the payload in three levels (a division maintained in the JSON format):

- The feed: envelops a set of resources
- An entry: envelops a single resource
- A property: is an individual information carrier in the context of a resource

The following sub-sections present the upcoming SData 2.0 changes for each level.

[2] This is a 'lesson learned' from the practical experiences with implementations of the standard. It was recognized that the objective needs of products and contracts, in combination with the underlying development tools, do not easily conform to the 1.x requirements. The relaxation reduces the implementation and maintenance effort of delivery teams.

Feed level

The following subsections discuss feed-level aspects of the SData 2.0 standard, namely:

- Namespaces
- Protocol-defined elements
- Links
- Categories

Namespaces

Namespaces are relevant only for the `atom+xml` format. The table below shows namespaces and their compliance level for SData 2.0:

Namespace	Compliance level
Http	MUST
Atom	MUST
SData	MUST
Xsi	MAY (meaningful only when schemas are involved)
Opensearch	MAY; meaningful only when paging is implemented
Sle	MAY; meaningful in a very reduced number of cases

Non-essential namespaces (sle, xsi, and others) should be used as dictated by the individual necessities of a scenario.

Protocol-defined elements

SData 2.0 recognizes two payload formats: `atom+xml` and `json`.

ATOM elements[3] are dictated by the ATOM specification and SData, being ATOM conformant, MUST require the presence of these elements in the `atom+xml` format. As only moderate benefits are derived in the SData from ATOM elements, the support is kept at a minimum. The SData 1.x definitions are found in section 3.2 Feed Elements.

In the JSON format, not being attached to a standard, SData keeps a free hand. Most of the ATOM elements deemed meaningful for JSON maintain similar names - these are preceded by a **$** (example: `title` in atom becomes `$title` in JSON).

The following table presents the SData 2.0 feed-level elements:

	SData 1.x	SData 2.0	SData 2.0 JSON	Notes

[3] These elements are hardly any use to us in practical terms

		ATOM		
id	MUST	**MUST**	NO	In JSON, this information is carried by the $url object (see below)
title	MUST	**MUST**	**MAY**	Just as a matter of good practice, encourage people to fill this
updated	MUST	**MUST**	**MAY**	Meaningful for syndication context but with little value for SData. In practice, we will not compute on every GET the updated value for the whole feed.
author	SHOULD	**MAY**	NO	A relaxation to the 1.x standard
summary	MAY	**MAY**	NO	To be specified only if value can be derived from the element
category	SHOULD	NO	NO	This is a candidate for **deprecation**
xml:base	NO	**SHOULD**	NO	Allows specification of relative URLs, thus greatly reducing the size of the payloads (see XMLbase specification)
$resources	NO	NO	**MUST**	The JSON object that envelopes individual resources
$baseUrl	NO	NO	**MAY**	The JSON pendant to the xml:base attribute
$url	NO	NO	**MAY**	The JSON pendant to the ID and Self ATOM links
$links	NO	NO	**MAY**	The JSON object encompassing the link elements of a feed
$diagnosis	NO	NO	**MAY**	The JSON pendant to the ATOM diagnosis element (see: sdata.xsd, and 3.10 Error Payload)

Links

Links are used to point to resources (data, services, etc.) related to an object. In the `atom+xml` format, links have a standardized format (see section 4.2.7 of the ATOM syndication format). In JSON, links are represented as sub-objects of the `$links` object mentioned previously.

In the SData 2.0 specification of any specific link is optional. Specific links may be required by a contract, the standard however does not impose any such requirements.

SData 2.0 recommends that the URL components of a link object (with the exception of the ATOM `id` element) be defined relatively. This is achieved by using:

- the `xml:base` attribute in the atom+xml format
- the `$baseUrl` name-value pair in JSON

The names of links are not standardized in the 1.x version of the SData standard. Although no standardization effort is undertaken by the SData 2.0 version, it is recommended that the IANA Link relations be consulted prior to defining a new link name. The following is a list of links as they emerged in examples of the 1.x version and their handling in the 2.0 version:

	SData 1.x	SData 2.0	Notes
self	MUST	MAY	It seems a little odd to include this at all as it is the same almost

			always identical to the ATOM ID element. However, existing implementations may rely upon its presence
first	MAY	MAY	Only in conjunction with a paged feed
next	"	"	"
prev	"	"	"
last	"	"	"
schema	MUST (XML)	MAY	Points to schema location of a resource kind
template	MAY	MAY	Points to template of a resource kind
post	MAY	MAY	Points to location of POST operation for a resource kind
queries	MAY	MAY	Points to location of relevant queries of a resource kind
service	MAY	MAY	Points to location of relevant services of a resource kind
batch	MAY	MAY	Points to location of relevant batching services of a resource kind

Categories

The Categories feature is a candidate for deprecation. Anyone with reasons to request support for this feature in SData 2.0 should do so on the corresponding Open team.

Entry level

The following subsections discuss Entry-level aspects of the SData 2.0 standard, namely:

- Protocol-defined elements
- Links

Protocol-defined elements

As mentioned in the corresponding section of the Feed Level chapter, for the `atom+xml` format the existence and nature of protocol defined elements results from the ATOM syndication definition.

In the JSON format the ATOM element names are prefixed by a **$** (example: `author` becomes `$author`). Additionally, as JSON had no attributes support, these are represented by objects or name-value pairs (see for example $key in the table).

The corresponding SData 1.x definitions are found in section 3.7 Typical Feed Entry. The following table presents the SData 2.0 entry-level elements:

Element	**SData 1.x**	**SData 2.0 ATOM**	**SData 2.0 Json**	**Notes**
id	MUST	**MUST**	NO	`$url` is the JSON counterpart
title	MUST	**MUST**	**MAY**	

updated	MUST	**MUST**	**MAY**	This is an alternate version recognition mechanism to the etag.
author	SHOULD	**MAY**	**MAY**	
summary	MAY	**MAY**	**MAY**	
category	SHOULD	NO	NO	This is a deprecation candidate
content	SHOULD	**MAY**	NO	
payload	MUST	**MUST**	NO	
etag	MAY	**MAY**	**MAY**	
$url	NO	NO	**MAY**	`$url` points to the entry. It is the JSON counterpart to the ATOM ID.
$key	NO	NO	**MAY**	`$key` contains the value of the primary key of an entry. It is the JSON representation of the property-level attribute key
$properties	NO	NO	**MAY**	Container for metadata associated with the entry
$links	NO	NO	**MAY**	Object containing links that present functional aspects of the resource (example: edit, lookup, create). They are to be understood as hypermedia controls
$diagnosis	NO	NO	**MAY**	The JSON pendant to the ATOM diagnoses element (see: sdata.xsd and 3.10 Error Payload)

Links

In the SData 2.0 specification of any specific link is optional.

The discussion on feed-level links holds true for entry-level links as well. A more comprehensive standardization on links is currently under consideration.

Property level

There are several xml attributes defined in the sdata.xsd document that may annotate a resource's properties; these remain valid in SData 2.0.

As JSON has no comparable mechanism, xml attributes are represented in JSON as name/value pairs:

atom+xml attribute name	JSON name	JSON Value type
key	`$key`	String
url	`$url`	String
uuid	`$uuid`	String
lookup	`$lookup`	String (*is a link located in the $links object*)
descriptor	`$title`	String
isDeleted	`$isDeleted`	Boolean
deleteMissing	`$deleteMissing`	Boolean
index	`$index`	Reserved

Requesting content by means of media type negotiation

SData 1.x relied solely on the `atom+xml` format. SData 2.0 introduces support of the JSON format on par with `atom+xml`, giving applications the freedom to operate in any one or both formats. A consumer can explicitly request one of the supported formats or alternatively, rely upon the default provider response.

Explicit request

The explicit request for a format is achieved by specifying the media type "`atom+xml`" or "`json`" within the Accept header of the request or as a query parameter as shown below:

Accept header	**Atom**	`Accept: application/atom+xml;vnd.sage=sdata`
	JSON	`Accept: application/json;vnd.sage=sdata`
Format URL parameter	**Atom**	`…?format=application/atom+xml;vnd.sage=sdata`
	JSON	`…?format=application/json;vnd.sage=sdata`

Default/implicit request

A contract can (and should) specify the default format it operates on. The default is chosen to service the majority of the incoming requests without additional specification from the consumer. The following defaults are suggested:

Media type	Defaults to
`Application/json`	`Application/json;vnd.sage=sdata`
`Application/xml`	`Application/atom+xml`
`No media type specified`	`Use the contract specified default`

Contracts not explicitly naming a default format are assumed to have the `application/atom+xml` default. This ensures consistency with the current SData 1.x implementations.

Operations

The SData 1.x. specification describes the following logical operations:

- 7 Read Operation
- 6 Queries
- 8 Create Operation
- 9 Update Operation
- 10 Delete Operation

SData 2.0 logical operations are implemented through HTTP verbs as shown in the table below:

SData operation	HTTP Verb	Compliance level
Read	GET	SHOULD

Update – full contents	PUT	MAY
Update – partial contents	PATCH	MAY
Create operation	POST	MAY
Delete operation	DELETE	MAY
Queries	GET	SHOULD: support of the basic query set. Properties usable in queries are at least: • sdata:key property if one is defined • sdata:uuid property if one is defined MAY for other properties

The PATCH support is a new addition to SData; it supplants the PUT verb usage for partial payloads. The usage of PUT for partial updates is **deprecated** with SData 2.0. Existing implementations may still use PUT, but should consider switching to PATCH in upcoming releases.

Authorization

SData 1.x discusses authorization aspects in the security section (5 Security) of the standard. **These remain valid in SData 2.0**.

SData 2.0 will provide means to integrate with SageID and will specify the needed support for single sign on. The relevant documents are currently under development.

Status and error codes

SData 1.x presents in section 3.10 Error payload the information exchange from provider to consumer. **These remain valid in SData 2.0**.

The JSON object structure for the diagnoses and diagnosis xml elements are described in the JSON documentation.