SData 2.0: Sage ID Integration

Version 1.0

1 Introduction

1.1 Background

SData [1] is a standards-based protocol used by many Sage products to share information and promote integration. SData is based on HTTP (Hypertext Transfer Protocol), the protocol that powers most of the internet traffic around the globe, and is suitable for use in Application Programming Interfaces (APIs), mobile applications, and for application integration.

SData is published under a Creative Commons Licence, and may be freely reused as a specification.

SData version 1.0 was published in 2010, and updated in 2011 to the current version, v1.1. This document, along with four others defines the next version of SData, 2.0, which focuses on simplifying the protocol and introduces full support for JavaScript Object Notation (JSON) [2], alongside the XML and Atom support introduced in SData v1.0.

The full set of documents defining SData 2.0 is:

"The underlying approach to evolving SData"	Outlines how the SData Working Group approached the task of updating the SData protocol while ensuring compatibility with implementations of version 1.0 and 1.1 already in use.
"SData 2.0 Core"	Defines the main elements of the SData protocol, explains how these elements are being updated (and in some cases, relaxed) for the 2.0 release, and outlines how JSON is being integrated into these elements.
"JSON formatted SData responses"	Defines the JSON format for SData content, focusing on structural aspects of content and representation.
"SData 2.0 Expressing metadata in JSON"	Builds on the JSON formatted SData responses to define how providers should express metadata in JSON.
"SData 2.0 and Sage ID"	This document. Specifies how Sage ID Authentication is handled in SData 2.0.

1.2 Overview

Sage ID is the user authentication service for Sage online products and services. Further information about Sage ID can be found online at http://docs.sso.sagessdp.co.uk (username: ssodocs, password: Q9VdcpfkWFbT).

This document specifies how Sage ID tokens should be used in conjunction with SData 2.0.

2 Using Sage ID with SData

Sage ID provides support for the authorization code grant type described in RFC 6749, "The OAuth 2.0 Authorization Framework" [3]. A client participating in this protocol receives a "bearer token" which is presented to a service as evidence of authorization. The way that the bearer token is transferred between the client and the service is defined in RFC 6750, "The OAuth 2.0 Authorization Framework: Bearer Token Usage" [4].

The Bearer Token Specification does not specify the contents of the token (this is intentionally left to be implementation specific), but it does specify the mechanisms for presenting access tokens in resource requests.

There are three main mechanisms for presenting access tokens:

- HTTP request header
- HTTP request entity-body
- HTTP URI query parameter

Only the first of these mechanisms is supported by the SData 2.0 Protocol.

2.1 HTTP request header

When sending the access request token in the Authorization header, the access token must be base64 encoded and preceded by the literal string "Bearer" (without the quotation marks):

```
GET /resource HTTP/1.1
Host: server.example.com
Authorization: Bearer vF9dft4qmT
```

2.1.1 Server Response

When an SData service receives a request via any of these mechanisms, it should:

- Base 64 decode the token
- Decrypt the decoded token using the service symmetric key (previously supplied when the service was registered with Sage ID)
- Validate the token according to the procedure illustrated in the Sage ID documentation and sample code

If token validation succeeds and the request is authorized, the service should fulfill the request.

2.2 Authorization Error Conditions

There are several authorization error conditions that the SData service MUST cater for:

2.2.1 Missing access token

No access token was included with the request.

In this case the SData service MUST return a HTTP 401 (unauthorized) status code with a WWW-Authenticate header but MUST NOT include any other error information:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate:  Bearer realm=”SageID”
```

Note that the WWW-Authenticate header used in this and all other authorization error conditions MUST use the auth-scheme value “Bearer” and a realm of “SageID”, as shown above.

2.2.2 Multiple access tokens

More than one access token was included with the request.

In this case the SData service MUST return a HTTP 401 (unauthorized) status code with a WWW-Authenticate header with an error parameter of “invalid_request” and an error_description parameter with a textual description of the cause of the error:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm=”SageID”,
error=”invalid_request”, error_description=”Multiple access
tokens were supplied.”
```

2.2.3 Invalid access token

An access token was included correctly with the request, but the provided token is malformed, expired or invalid for other reasons.

In this case the SData service MUST return a HTTP 401 (unauthorized) error code and a WWW-Authenticate header with an error parameter of “invalid_token” and an error_description parameter with a textual description of the cause of the error:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate:  Bearer realm=”SageID”,
error=”invalid_token”, error_description=”The access token was
expired.”
```

The error_description parameter MUST NOT include detailed technical information about the cause of the error which might allow a malicious user to ascertain details of the underlying implementation or to conduct attacks against the cryptography used to decrypt and validate the access token. It is RECOMMENDED that, if the token is invalid for any reason other than expiry, the error_description parameter should simply state that the access token was malformed.

2.2.4 Access token with insufficient scope

The provided access token is valid but the request requires higher privileges than encoded in the access token.

In this case the SData service MUST return a HTTP 401 (unauthorized) error code and a WWW-Authenticate header with an error parameter of “insufficient_scope” and an error_description parameter with a textual description of the cause of the error:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate:  Bearer realm=”SageID”,
error=”insufficient_scope”, error_description=”The access
token did not contain the required permissions.”
```

Note:
Use of the “scope” parameter is not covered in the SData Protocol’s integration with Sage ID. According to [3] and [4], values of the scope parameter are defined by the authorization service (Sage ID in this case), rather than by the resource owner.

3 Compliance

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119, “Key words for use in RFCs to Indicate Requirement Levels” [5].

References

Number	Title	Version	Date	Author
1	**SData** *Welcome to SData*	1.1	2011	Sage Group plc
2	**RFC 4627** *The application/json Media Type for JavaScript Object Notation (JSON)*	Informational	July 2006	Internet Engineering Task Force
3	**RFC 6749** *The OAuth 2.0 Authorization Framework*	Proposed Standard	October 2012	Internet Engineering Task Force
4	**RFC 6750** *The OAuth 2.0 Authorization Framework: Bearer Token Usage*	Proposed Standard	October 2012	Internet Engineering Task Force
5	**RFC 2119** *Key words for use in RFCs to Indicate Requirement Levels*	Best Current Practice		Internet Engineering Task Force