BLEST (Batch-able, Lightweight, Encrypted State Transfer) - A modern alternative to REST

I. Introduction

REST (Representational State Transfer) is a widely adopted architectural pattern for designing networked applications. It has played a crucial role in the development of modern web services by providing a simple and scalable approach for building APIs. However, REST has limitations:

1. Over-fetching and under-fetching of data
2. Lack of uniformity in data representation (JSON, XML, etc.)
3. Performance overhead incurred by verbose HTTP headers and the multiple requests needed to perform simple operations

With the increasing complexity and scale of modern applications, there is a growing need for an alternative to REST that can address its limitations while providing the benefits of simplicity, scalability, and interoperability. This white paper introduces a promising alternative that aims to overcome those limitations, making possible the next generation of connected applications.

II. Overview

BLEST (Batch-able, Lightweight, Encrypted State Transfer) builds upon the foundation laid by REST and makes adjustments to improve performance, reduce resource consumption, and better serve the needs of modern application developers. Key features and benefits include:

1. Built on JSON
   BLEST uses JSON (JavaScript Object Notation – a popular data interchange format for web APIs) for all payloads which creates uniformity in data representation and allows developers to leverage existing parsing libraries.
2. Request Batching
   BLEST requests are batch-able which allows an application to perform multiple operations in a single HTTP request. This saves bandwidth, reduces latency, and lowers performance overhead on the server.
3. Compact Payloads
   BLEST specifies a clear and succinct payload structure that is as easy to read and write as it is efficient to send.
4. Selective Returns
   BLEST defines an optional method for specifying exactly which data the server should return, solving the over-fetching problem.
5. Single Endpoint
   BLEST eliminates the use of URI-based routing and instead uses strings in the payload to route the included request(s). This reduces complexity and prevents sensitive data or traffic patterns from leaking through request URIs.
6. Fully Encrypted
   BLEST requires the use of end-to-end encryption such as TLS (HTTPS).

III. Technical Details

BLEST is designed for use over HTTPS, but since all relevant data is JSON contained in the payload, it could theoretically be used over any protocol that supports JSON payloads. The JSON request and response formats are illustrated below:

Request Payload

A JSON array of request arrays comprised of a unique request ID, a route string, a parameter object (optional), and a return selector array (optional)

```
[ [ “abc123”, “math", { "operation": “divide", "dividend": 22,
"divisor": 7 }, [ “status”, [ “result”, [ “quotient” ] ] ] ] ]
```

Response Payload

A JSON array of response arrays comprised of the request ID, the route string, a result object (or null if error), and an error object (if applicable)

```
[ [ “abc123", “math", { "status": "Successfully divided 22 by 7”,
"result": { "quotient": 3.1415926535 } }, { "message": "If there was
an error you would see it here” } ] ]
```

Reference implementations and examples are available at the GitHub link below. At the time of writing there are server-side libraries for NodeJS and Python and a client-side library for React.

IV. Summary

BLEST offers a straightforward, standardized protocol for inter-application communication that builds from the foundation laid by REST. It solves many of the problems plaguing applications today such as over- and under-fetching, lack of uniformity, and performance overhead. It also reduces API complexity, increases flexibility, and improves security. In doing so, BLEST meets the challenge of connecting the increasingly complex and scalable applications of the future.

Developers interested in improving the performance, efficiency, flexibility, and security of their applications are encouraged to familiarize themselves with the protocol through the reference implementations and examples (GitHub link below) and try it for themselves on a test project.

V. Resources

BLEST reference implementations and examples: https://github.com/jhuntdev

Comments, questions, and support requests may be directed to blest@jhunt.dev

Published June 2, 2023 | Updated June 16, 2023