Common Component Specification Proposal

Date	2015-08-17
Spec Title	**Cloud Data Object (CDO) Protocol** (CCS-CDO-PRO)
Author	Peter Judge
Status	New
Version	1.0.0

Executive Summary

This document contains a proposal to the Common Component Standards (CCS) Steering Committee to form a new project team whose goal it is to produce a specification for a **Cloud Data Object (CDO) Protocol** to define a protocol for messages sent between a client and service endpoints. The protocol describes how clients should request resources to be fetched or updated, and how servers should respond to such requests.

The following sections of the document describe in general terms what the component is, the benefits it provides to application developers, and what the team's deliverables would be.

Component Description

This component formalizes the structure of the messages that are used for communication between a client and a typically-REST-based service.

This component describes a clear-text format that provides elements that describe the resource data in the message. This format includes:

- *Protocol metadata*; specifically a protocol version,
- *Error data*, from errors and exceptions that arise from the execution of the request, whether from the service or infrastructure where possible,
- *Message metadata* such as message identifiers, message context and related data,
- *Data schema* that provides a description of the resource, and
- *Resource data*: the resource that's being fetched or modified

Furthermore, the component must provide a mechanism for reducing unwanted network traffic (both in size and number of requests). This may be done via sending

bundled/aggregate messages and/or allowing certain elements to be optional. For instance, not all messages contain error data.

This specification will describe a message protocol that operates within the existing transport protocols such as HTTP and does not replace them. This component does not prescribe a model (such as request/response or server push) for initiating a message conversation.

Benefits and Use Cases

The primary use-case for this component is communication between clients that implement the CDO API specification and services described by the CDO Catalog. These messages are transported over HTTP from a browser-based JavaScript client to a RESTful server, and back.

This component can be used for any messages that travel between clients and servers, since it is transport-protocol agnostic.

The primary benefit of this component is that it removes the need for implementers of services and clients to know anything about the other end of the message conversation. This allows for easier testing of clients and servers (without requiring the other) and independent development of clients and server technologies.

In HTTP terms, a client may send a compliant message to a server with an appropriate Content-Type header, and know that the server will be able to determine whether it can service the request (via *200/OK*) or not (via a *415/ Unsupported Media Type* HTTP status code) or some other transport-specific return status.

Related / Dependent Common Component Specifications

This component does not depend on any other components, but since the primary use-cases for this component are likely to be service endpoints as described by a CDO Catalog, and implementations of the CDO API, the use-case of those components will inform and guide this component.

Project Team Requirements

A CDO Protocol team has not yet been formed. The team's formation will commence upon acceptance of this standards specification proposal by the CCS Steering Committee. The goal of this project team will be to submit a Community Review Draft of the Specification to the CCS Steering Committee within 90 days after the team is formed. If the specification is accepted, it will be published to the entire CCS participant list for review.

Deliverables

- A formal specification of the protocol, including MIME type and schema. This includes the possibility of using an existing/predefined protocol and contributing to such a project.
- Samples of client creation & consumption of messages
- Samples of server creation & consumption of messages