Online Book Catalogue Protocol – OBCP/1.0

Abstract

The Online Book Catalogue Protocol (OBCP) is a simple protocol used to communicate book entity data between a client and server. This request for comments document describes; an overview of the protocol, the format and order of client requests and server responses, errors and error handling, border-case behavior, and synchronization policies.

Table of Contents

Introduction

The Online Book Catalogue Protocol (OBCP) describes a system for communication between an online book catalogue client and server. It is based on the request/response paradigm popularized by the Hypertext Transfer Protocol (HTTP) and uses the Transport Control Protocol (TCP) for

transmitting messages. The client connects with the server, sends a request and awaits a server response. Every request given by the client involves the manipulation of a book entity. Clients can issue requests to the server to get, submit, and remove books.

Description of Protocol

The client communicates with the server over a TCP connection. The catalogue server remains on with a stable IP address and port 80 open, waiting for clients to connect. Multiple clients may establish connections with the server at once. After the final request has been sent by a client and responded to by the server the TCP connection between client and server will be closed by both parties. In the event that the server shuts down, all book entities previously stored are cleared from memory (no persistence on server).

Book entities passed by the book catalogue client and stored on the online book catalogue server are defined as plain text (ASCII) objects consisting of a book title, author, and location. Memory constraints for book objects will depend on the specific implementation of the book catalogue on the server. As an example, one book instance can be defined as:

Introductory Computer Forensics
Xiaodong Lin
Reynolds Bldg, 2210

In this example the book title is labelled *Introductory Computer Forensics*, the book author is *Xiaodong Lin*, and the location where the book is held is *Reynolds Bldg, 2210.*

The online book catalogue server is stateless and thus does not take into consideration a client's previous requests when handling a particular request. Instead it delivers a response based on the information given in the request, internal book catalogue functionality, and internal catalogue inventory (book entities stored).

A client request is said to be **valid** when it matches the expected format for its corresponding request.

To communicate with the online book catalogue server, the client sends book requests as outlined in the following section. Assuming the request is valid the server will respond as outlined below. Invalid requests will be handled according to the error handling specifications.

Each client request and server response include a header and body. The only information in the client request header is the request/query name (submit, get, remove). The client request body includes the fields required for the query type with their corresponding values. In a server response the response header includes a status code and information about the records modified, if any. The data set inside the server response body will be specific to the client request. It may include book entity data or an error message if the request was invalid.

There may be a scenario in which multiple clients attempt to access or modify the same book data on the server. Server behaviour for edge cases can be found in this section. Clients making

concurrent requests for the same book entity will be dealt with according to the synchronization policies.

Client/Server Operations

The online book catalogue protocol performs under the request/response paradigm. A client request **must** be sent before a server response is returned. The server will never initiate contact.

Client Requests

Client requests for each of the book catalogue queries (submit, get, remove) start with the name of the desired query type followed by a newline character. The remaining content to be sent to the online book catalogue server will be specific to the query. For each field defined in the query, a space needs to be appended before defining the field value. After the last piece of book data is defined in the query the client must provide an additional newline to mark the end of the query and send the request.

SUBMIT Request

Format

SUBMIT\n
TITLE_{Book Title}**\n**
AUTHOR_{Book Author}**\n**
LOCATION_{Book Location}**\n**
\n

Action

This request will submit a valid book entity with title, author, and location to the online book catalogue server. All book entity fields must be specified to send a valid submit request.

If the book title, author, and location in the submit request match a book found on the server the query will do nothing.

GET Requests

There are three different types of client get requests for a book entity based on the fields provided in the query. The client may request either all books with a specific title, all books from a specific author, or all books with a specific title from a specific author.

GET BY TITLE

Format

GET\n
TITLE_{Book Title}**\n**
\n

Action

This request will attempt to retrieve records of all book entities from the online book catalogue server that have the title specified in the query.

GET BY AUTHOR

Format

GET\n
AUTHOR_{Book Author}**\n**
\n

Action

This request will attempt to retrieve records of all book entities from the online book catalogue server written by the author specified in the query.

GET BY TITLE AND AUTHOR

Format

GET\n
TITLE_{Book Title}**\n**
AUTHOR_{Book Author}**\n**
\n

Action

This request will attempt to retrieve records of all book entities from the online book catalogue server that have the book title specified and have been written by the author specified in the query.

REMOVE Request

Format

REMOVE\n
TITLE_{Book Title}**\n**
AUTHOR_{Book Author}**\n**
\n

Action

This request will attempt to remove all instances of books from the online book catalogue server with the specified title written by the specified author. Only the book title and author need to be specified.

If the book title and author do not match a book found on the server then the query will do nothing.

Server Responses

Server responses to client requests follow a similar structure to those defined above except they have a response code in the header specifying the status (result) of the most recent request.

As with the client request, the exact content of the response message will be determined by the type of request received and the content stored on the server.

Response Codes
For every response provided by the server the server will send a status describing its understanding of the previous request. There are three different status codes; 100 OK, 400 MALFORMED PAYLOAD, and 600 SERVER ERROR.

100 OK
This status code signifies that the book catalogue server has understood the client's request. The data supplied in the request query was up to OBCP standard and operations on the server-side were performed as usual.

400 MALFORMED PAYLOAD
This status code signifies an error made in the client request. As such there was an error parsing the request.

600 SERVER ERROR
This status code signifies an unexpected error coming from the server. These errors typically come as a result of an application-specific exception.

SUBMIT Response
Format

100 OK – {Book Count} **Record(s) Updated\n**
\n
{Book Title}**\n**
{Book Author}**\n**
{Book Location}**\n**
\n

Action
This response demonstrates a successful submit request. Online book catalogue server responds with header including a 100 OK status code and the number of records updated (if any). In the response body the server shows the new book entity stored.

GET Response
Format

100 OK – {Book Count} **Record(s) Found\n**
\n
{Book Title}**\n**
{Book Author}**\n**
{Book Location}**\n**
\n
{Book Title}**\n**
{Book Author}**\n**
{Book Location}**\n**

Action
This response demonstrates a successful get request. Online book catalogue server responds with header including 100 OK status code and the number of book records retrieved that matched the request criteria. In the response body the server shows the book entity or entities found with newlines separating each entity.

REMOVE Response

Format

100 OK – {Book Count} **Record(s) Removed\n**

Action
This response demonstrates a successful remove request. Online book catalogue server responds with a header including a 100 OK status code and the number of book records removed that matched the request criteria.

Errors and Error Handling

Errors may occur on both the client and server side.

Client-side Errors and Server Responses

Invalid Request

Error that occurs when client request contains incomplete or otherwise invalid query data.

Example cases:

- Request doesn't include base query label (submit, get, remove).
- Request doesn't include required book fields for query.
- Request doesn't include required book field values for query.
- Request has required book fields in incorrect order.

For all of these cases the online book catalogue server returns a response with a status code of 400 MALFORMED PAYLOAD. The response body for a response where the server receives a malformed payload is specific to the type of request sent by the client (submit, get, remove). In the event that the server is unable to determine the expected request format for the request sent, the server will simply respond with the status code and the payload received. This is done as the server doesn't have enough information to determine what the client is requesting.

Server Response to SUBMIT – Client Error

Format

400 MALFORMED PAYLOAD – {Book Count} **Record(s) Updated\n**
\n
Payload Received:\n
{Client Payload}**\n**
\n
Expected:\n
SUBMIT\n
TITLE_{Book Title}**\n**

AUTHOR_{Book Author}**\n**
LOCATION_{Book Location}**\n**

Action
This response demonstrates an unsuccessful submit request based on an invalid client request payload. Online book catalogue server responds with a 400 MALFORMED PAYLOAD status code in the header and the number of records updated (zero). In the response body the server shows the payload given by the client followed by the expected format for a submit request.

Server Response to GET – Client Error

Format

400 MALFORMED PAYLOAD – {Book Count} **Record(s) Found\n**
\n
Payload Received:\n
{Client Payload}**\n**
\n
Expected:\n
(1)
GET\n
TITLE_{Book Title}**\n**
\n
OR\n
\n
(2)\n
GET\n
AUTHOR_{Book Author}**\n**
\n
OR\n
\n
(3)\n
GET\n
TITLE_{Book Title}**\n**
AUTHOR_{Book Author}**\n**
\n

Action
This response demonstrates an unsuccessful get request based on an invalid client request payload. Online book catalogue server responds with a 400 MALFORMED PAYLOAD status code and the number of records found (zero) in the header. In the response body the server shows the payload given by the client followed by the 3 possible expected get request formats.

Server Response to REMOVE – Client Error

Format

400 MALFORMED PAYLOAD – {Book Count} **Record(s) Removed\n**
\n

Payload Received:\n
{Client Payload}**\n**
\n
Expected:\n
REMOVE\n
TITLE_{Book Title}**\n**
AUTHOR_{Book Author}**\n**
\n

Action
This response demonstrates an unsuccessful remove request based on an invalid client request payload. Online book catalogue server responds with a 400 MALFORMED PAYLOAD status code and the number of book records removed that matched the request criteria in the header. In the response body the server shows the payload given by the client followed by the expected format.

Server-side Errors

Server Error

Error that occurs on server-side due to some application-specific error. When processing a given request if the online book catalogue server encounters an error resulting from memory/indexing issues the server will return a status code and include the propagated message in the response body.

Server Response to Application Error

Format

600 SERVER ERROR\n
\n
{Error Message}**\n**
\n

Action
This response demonstrates when a server error occurs on any kind of request (submit, get, remove). Online book catalogue server responds with a 600 SERVER ERROR status code and delivers the error message propagated from the application.

Border-cases Behaviour

There are some scenarios in which handling on the server may be performed in multiple ways. One instance of this is when a client sends multiple submit requests for the same book entity. Another is when a client sends a get request for a book entity that does not exist. Similarly, a client may send a request to remove a book entity that does not exist. Finally, a client may send a request to remove a book entity which has multiple locations.

Duplicate Book on Submit Request

There are 2 cases to be handled by the server in the event of a duplicate client submit request.

The first is when the book entity described in the client request body matches all the fields (title, author, and location) of a book entity found by the server. Here the correct functionality for the online book catalogue server would be to simply do nothing and respond with a **100 OK** status code including **0 Records(s) Updated** in the response header. The previously created book record will be sent in the response body.

The second case is when the book entity described matches the title of a book entity found on the server but doesn't match the book author and/or location. In this scenario the correct action for the book catalogue server to take would be to create a new book entity and return a status code of **100 OK** followed by **1 Record(s) Updated** in the response header and the newly created record in the response body.

Book Does Not Exist on Get Request
If the client sends a get request for a book entity that does not exist on the online book catalogue server, the server should respond with a simple **100 OK** status code followed by **0 Record(s) Found** appended in the response header. The response header for this response will inform the client that their request was successful, and they didn't generate any results with their search. The response body will be empty.

Book Does Not Exist on Remove Request
If the client sends a remove request for a book entity that does not exist on the server, the server should respond with a simple **100 OK** status code followed by **0 Record(s) Removed** in the response header. This will inform the client that the operation went through, but no records were changed. The response body in this response will be empty.

Multiple Books at Different Locations on Remove Request
There is one case where a server response for a remove request may be ambiguous to a client. This is when the online book catalogue server finds multiple book entities with the same title and author stored at different locations. Recall that for a submit request, the client may submit a duplicate book entity so long as it has a different book location from the one found on the server. Example for 2 duplicate books found:

Introductory Computer Forensics	**Introductory Computer Forensics**
Xiaodong Lin	**Xiaodong Lin**
Reynolds Bldg, 2210	**McLaughlin Library, Floor 3, E8**

In this case the server will remove both instances of *Introductory Computer Forensics* by Xiaodong Lin. The server will respond with a **100 OK** status code followed by **2 Record(s) Removed** in the response header. As with other valid remove responses the response body will be left empty.

Synchronization Policies

The online book catalogue server may maintain up to 5 client connections simultaneously. In the event that multiple clients attempt to access or modify the same resource at once the client who was first to issue a request will have exclusive access to the resource.

In this scenario this client locks the aforementioned book entity while the other clients stay in a wait state. Once the client with the lock on the resource finishes executing their request, the client who is next in line (has the oldest request) will then lock the resource and have the server process its request.

This type of ordering is called first-in-first-out and the process is repeated until all client requests for the resource have been handled by the server.