AO Open Services Drivers for UI Support using JSON (V4.0)

Introduction

This definition document outlines the requirements for the submission of a request to an AOO Services Driver and the subsequent responses which can be expected.

It must be made clear at this point that the servers, as created for and by the AOO Generator, are completely stateless, and as such all variables required for subsequent use by the client instance must be stored in the client.

Endpoint Definition

The Endpoint (URL) starts with https://..... followed by an address defined within the APACHE configuration files to direct the request to the correct location, schema, and version. E.g. https://tembo72.tembotechlab.com/aoo/v1/.......

The 10-character (max) Id. following the last / is the name of the services driver compiled on the IBMi, providing the required service, and must conform to IBMi program naming requirements. It can be written in any ILE language, but all AOO servers are provided as RPG IV+/ILE.

Environment Header Variables

Request-Method: POST

This must always be POST. Any other value will cause the service drivers to return an error.

Content-Type: application/json; charset=utf-8

All AOO service programs are provided with UTF8>Local>UTF8 conversion procedures for the payload. Any other CCSID's may be used but procedures for the conversion procedures need to be included in the AOO service program.

Content-Length: nnnn

Some services do not require a JSON body to be sent and in these cases the content length must be 0. If the services program is not returning any JSON body the Content-Length will be set to 0.

HTTP-Authorization: Bearer x(44)

The x(44) is a 44-byte base64 encoded string, unique to the user, and must be present in the environment header of ALL incoming service requests, except LOGON. (See below).

X-Authorization-Token: x(44)

The x(44) is a 44-byte base64 encoded string, unique to the user, and is returned in the environment header of ALL responses returned by the service program, except the LOGOFF. (See below).

Status: 401 xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

This environment variable is returned by the service program if the HTTP-Authorization variable fails the profile validation routine for any reason. The x's are an error related text string. This variable will always contain the standard "Failure Response" message as the JSON body. (See below).

Protocol for JSON Payload

This definition document defines the protocols and construction of the JSON client requests to acquire information from the requested service, as well as the format and construction of the JSON responses from the service to the client.

This set of Request/Response protocols covers the most common requirements for Browse, Read, Edit, Add and Delete (BREAD) actions. Additional requirements will be added to cater for other communication requirements, as necessary.

Rules for KEY/VALUE Pairs

All requests and responses are made up of JSON objects {} containing Key/Value pairs. The "Key" and "Value" are separated by a colon (:) and Key/Value pairs in an object are separated by a comma (,). In some cases, the "Value" can be another JSON object {}.

KEYS:

Keys/Names in a request string may be double-quoted or not and can be supplied in lower, upper or mixed case. Keys/Names in a response string will always be double-quoted and in capital letters.

The Key/Name defined in the Key/Value pairs, if not a pre-defined constant, must be the correctly spelled column names of the rows "system" name. These names can be any case and quoted or unquoted but must be spelled correctly.

VALUE:

Character values may be double-quoted or not and the content may be in any combination of case. However, if a value is unquoted then all alphabetic characters will be translated to upper case. To retain the supplied case of a value it must be double quoted. In addition, if the character string contains special characters, then the string MUST be double quoted (") else unpredictable results can occur.

Numeric values must always be unquoted, as per the JSON best practice.

Services Provided

The following services are provided by the current release of AOO Open

Logon to AO Open (LOGON)

To use any of the AO Open functionality it is required to first "Logon" to the IBMi using a conventional IBMi User Id & Password which has been previously defined to the IBMi. The following LOGON function must be the first JSON string sent by a new user.

{"USERID":"userid","PASSWORD":"password","TIMEOUT": nnnn}

The TIMEOUT value can be from 1 to 3600 seconds. No TIMEOUT keyword or a value of 0 selects the default of 3600 seconds (1 hour).

Success Response

The environment header will contain an environment variable of "X-Authorization-Token:" followed by a space and then a 44-byte BASE64 encoded "TOKEN". There will be no JSON payload returned in the message body.

Failure Response

The Environment header will contain "Status: 401 xxxxxxxxxxxxxxxxxx", where the x's will be error-relevant text and the message body will be:

{"ERROR":["MsgId - MessageText", "MsgId - MessageText","…."]}

Logging Off AO Open (LOGOFF)

A TOKEN which is unused for the timeout period specified will automatically be invalidated, but it is advised that a user logs out at the end of a session, as good practice.

As with all requests after LOGON, the LOGOFF request must contain the current profile token in the HTTP_AUTHORIZATION variable of the environment header.

No JSON payload is required.

Success Response

On a successful response the environment header will be returned without the “X-Authorization-Token” variable no JSON statement will be returned in the message body.

Failure Response

As for LOGON.

Reset Authorization (AUTHRESET)

This service is used to reset the authority timeout counter which has a maximum of 1 hour from LOGON. The request must contain the HTTP-Authorization: Bearer x(44) environment variable containing a valid, unexpired token. If the token is expired or invalid the standard Failure Response will be returned from the service.

A JSON body may optionally be included in the request to set or change the timeout counter, as follows.

{“TIMEOUT”:nnnn}

If the body is not present or nnnn=0 the timeout will be set to 3600 seconds (1 hour).

Success Response

The environment header will contain an environment variable of “X-Authorization-Token:” followed by a space and then a 44-byte BASE64 encoded “TOKEN”. This will be the new, valid token for the user.

There will be no JSON payload returned in the message body.

Subsequent Requests

All JSON requests following a successful logon must contain a variable “HTTP-Authorization: Bearer xxxxxxxxxxx” in the environment header, where xxxxxxxxxx is the TOKEN returned by the logon sequence.

The requested service is identified by the Service Id. in the endpoint definition.

A request that contains an invalid or expired TOKEN will return the same failed response as that for the LOGON action.

Failure Response for All Requests:

The following JSON string will be the standard response for all failed requests.

{"ERROR":["MsgId - MessageText", "MsgId - MessageText","…."]}

The value of the “ERROR” key will be an array of messages separated by commas. Each message received in response to an error will be an occurrence in the array. These messages will be in reverse order, i.e. the last message, usually the final *ESCAPE message, will appear first in the response

Record Selection

This function will return a record set identified by the related service. Current services provided are related to a single DB file. The Service Id. is the “System” file name.

Request:

"ACTION":"SELECT",	Mandatory for Record Selection.
"COUNT":"Y",	Request to count the number of records defined by the "COLUMNS" selection. "OFFSET" and "LIMIT" keys are ignored, and may be omitted, when "COUNT" is used.
"OFFSET":nnnnn,	The row number of the row in the record set to start from, defined by the "COLUMNS" criteria. Leaving out the "OFFSET" or setting the value to 0, indicates to start at the beginning of the record set.
"LIMIT":nnnnn,	The maximum number of records to be returned from the "SELECT". Leaving out the "LIMIT" or setting the value to 0 indicates that the default limit of 200 rows will be returned, if available.
"COLUMNS":{	Specifies the columns used to filter the records. Enter only the columns required to be used for the filter. Any columns in the row can be used as a filter column. If no filtering of records is required, then leave out the "COLUMNS" key.
"name":"value"	A repeating key/value pair identifying the name and value of the column to be used in the filter. If more than one key/value is required for the filter, they must be separated by a comma(,). The percentage (%) sign may be used as a wildcard character in character and date values, to select a value beginning with, ending with, or “containing” a partial value. Wildcard characters used in numeric values require that the entire value be double quoted as it is treated as character
}	Indicates the end of the “COLUMNS” object.
}	Indicates the end of the request object.

Example:

{"ACTION":"SELECT", "COUNT":"Y", "OFFSET":nnnnn, "LIMIT":nnnnn, "COLUMNS":{"name":"value", "name":"value", . . . }}

Success Response for "COUNT":"Y"

{"COUNT":nnnnn}

Success Response without "COUNT"

{"ROWS":{"name":"value", "name":"value", x once for each column in the row}, {"name":"value", "name":"value", x once for each column in the row},{ . . . repeated until no. of rows ="LIMIT" . . . }}

Adding new row

All mandatory columns must be provided, as "name":"value" pairs, in the request below. Optional columns may be left out and other columns, provided by the database will be ignored.

Key:Value pairs may be provided in any order. Columns which are left out of the request, other than the optional ones, will be provided with standard default values which may or may not cause validation errors on insert.

NOTE: Values must ALWAYS be provided for all key elements of the row, except for those key values which are created internally by the database. Failure to do so will result in errors or unpredictable results.

Request:

{"ACTION":"INSERT", "COLUMNS":{"name":"value", "name":"value", . . . }}

Success Response:

The "Success" response, below, will echo all the values of the fields, in the column order, including those assigned by the database.

{" COLUMNS ":{"name":"value", "name":"value", x once for each column in the row}}

Updating an existing row

Updating an existing row is like the "Insert". In addition, only fields which to be updated need to be included. Fields not included will remain unchanged or defaulted by the database.

NOTE: Values must ALWAYS be provided for all key elements of the row. Failure to do so will result in multiple rows being updated, using the key values supplied. If no key values are provided an error will be returned and the update will fail.

Request:

{"ACTION":"UPDATE", "COLUMNS":{"name":"value", "name":"value", . . . }}

Success Response:

The "Success" response will echo the full updated row as in the "INSERT" action above. If only partial key values were provided then all the updated rows will be echoed, in the same format as the "SELECT" response.

Deleting an existing row

When requesting a delete, only the columns making up the keys need to be supplied. For files with multiple key columns, all keys must be specified, separated by a comma. Failure to do so will result in multiple rows being deleted, using the key values supplied. If no key values are provided an error will be returned and the update will fail. The keys can be supplied in any order.

To allow for a "Delete" confirmation, use the "Display" action to display the record and allow for confirmation or cancellation before requesting the delete.

Request:

{"ACTION":"DELETE, " COLUMNS ":{"name":"value", "name":"value", . . . }}

Success Response:

The "Success" response will echo the full deleted row as in the "Insert" action above. If only partial key values were provided then all the deleted rows will be echoed, in the same format as the "SELECT" response.

Display all columns in a row.

Files often have more fields than it is comfortably possible to display horizontally across the screen in a filtered list. This "Display" request returns ALL fields defined in a table row for a single key.

When requesting a display, only the columns making up the keys need to be supplied. For files with multiple key columns, all keys must be specified, separated by a comma. The keys can be supplied in any order.

Request:

{"ACTION":"DISPLAY, " COLUMNS ":{"name":"value", "name":"value", . . . }}

Success Response:

The "Success" response will echo the full row as in the "Insert" (Add) action above.