# 3 Implicit Parameters

This chapter describes the implicit parameters used in REST service handlers that are not explicitly declared. Oracle REST Data Services (ORDS) adds these parameters automatically to the resource handlers.

ORDS also supports, under certain conditions, automatic binding of the following:

* Query parameters
* Form data
* JSON objects

When query parameters are provided, they are always automatically bound by Resource Handlers. Whereas automatic binding behavior of form data and JSON objects are dependent on the following two factors:

1. Where and how the :body, :body\_text, and :body\_json implicit parameters are used, *and*
2. The media- or MIME type used:
   * application/x-www-form-urlencoded
   * application/json
   * multipart/form-data *with a single file*
   * multipart/form-data *with multiple files*

**NOTE:** Sections **3.1.1 About the :body parameter**, **3.1.2 About the :body\_text parameter**, and **3.1.3 About the :body\_json parameter** will cover in detail automatic binding behavior under various conditons.

## 3.1 List of Implicit Parameters

The following table lists the implicit parameters:

Note:Parameter names are case sensitive. For example, :CURRENT\_USER is not a valid implicit parameter.

Table 3-1 List of Implicit Parameters

| Name | Type | Access Mode | HTTP Header | Description | Introduced |
| --- | --- | --- | --- | --- | --- |
| :body | BLOB | IN | N/A | Specifies the body of the request as a temporary BLOB. | 2.0 |
| :body\_text | CLOB | IN | N/A | Specifies the body of the request as a temporary CLOB. | 18.3 |
| :body\_json | CLOB | IN | N/A | Specifies the body of the request as a temporary CLOB in JSON format. | 24.1 |
| :content\_type | VARCHAR | IN | Content-Type | Specifies the MIME type of the request body, as indicated by the Content-Type request header. | 2.0 |
| :current\_user | VARCHAR | IN | N/A | Specifies the authenticated user for the request. If no user is authenticated, then the value is set to null. | 2.0 |
| :forward\_location | VARCHAR | OUT | X-ORDS-FORWARD-LOCATION | Specifies the location where Oracle REST Data Services must forward a GET request to produce the response for this request. | 18.3 |
| :fetch\_offset | NUMBER | IN | N/A | Specifies the zero-based offset of the first row to be displayed on a page. | 18.3 |
| :fetch\_size | NUMBER | IN | N/A | Specifies the maximum number of rows to be retrieved on a page. | 18.3 |
| :page\_offset | NUMBER | IN | N/A | Specifies the zero based page offset in a paginated request. Note: The :page\_offset parameter is deprecated. Use :row\_offset parameter instead. | 2.0 |
| :page\_size | NUMBER | IN | N/A | Specifies the maximum number of rows to be retrieved on a page. Note: The :page\_size parameter is deprecated. Use :fetch\_size parameter instead. | 2.0 |
| :row\_offset | NUMBER | IN | N/A | Specifies the one-based index of the first row to be displayed in a paginated request. | 3.0 |
| :row\_count | NUMBER | IN | N/A | Specifies the one-based index of the last row to be displayed in a paginated request. | 3.0 |
| :status\_code | NUMBER | OUT | X-ORDS-STATUS-CODE | Specifies the HTTP status code for the request. | 18.3 |

### 3.1.3 About the :body\_json parameter

The :body\_json implicit parameter can be used with POST Resource Handlers to receive the contents of the request body as JSON object. This allows Resource Handlers to directly reference JSON properties (i.e., {"key": "value"} pairs).[[1]](#footnote-20)

Additionally, the :body\_json implicit parameter can be used when form data and one or more files are included in multipart/form-data POST requests. Form data, bound to the :body\_json implicit parameter, continues to be received as a JSON object while [one or more] files can be processed with the ORDS.BODY\_FILE\_COUNT LOOP function and the ORDS.GET\_BODY\_FILE procedure.

Similar to the :body and :body\_text implicit parameters, when the:body\_json implicit parameter is included in a Resource Handler, *it must be invoked* in order to be used. The :body\_json parameter can be invoked in various ways, such as:

* The DBMS\_OUTPUT package such as dbms\_output.put\_line(:body\_json);
* The hypertext procedures (htp) and functions (htf) packages, such as in htp.print(:body\_json);
* Assigning the :body\_json implicit parameter as variable, e.g. l\_body\_json := :body\_json;

#### Scenarios for using :body\_json

The below table summarizes the possible scenarios where the :body\_json implicit parameter can be used. When form data in the POST body request is to be received as a JSON object, the :body\_json implicit parameter should be used for the MIME types seen below. Pay special attention to the multipart/form-data request in cases where you intend to send 1 or more files in a request.

MIME type

application/x-www-form-urlencoded

application/json

multipart/form-data

POST body contents

Form data (when in x-www-form-urlencoded format)

-

-

-

Form data (as JSON object)

:body\_json

:body\_json

:body\_json

≥ 1 file/s included

-

-

ORDS.BODY\_FILE\_COUNT LOOP& ORDS.GET\_BODY\_FILE

#### Example

A table (DEMO\_TABLE) has been created with the following attributes:

CREATE TABLE DEMO\_TABLE   
 (   
 ID NUMBER (\*,0) GENERATED BY DEFAULT AS IDENTITY   
 ( START WITH 1 CACHE 20 ) NOT NULL ,   
 FILE\_NAME VARCHAR2 (200) ,   
 FILE\_BODY BLOB ,   
 CONTENT\_TYPE VARCHAR2 (200) ,   
 FILE\_VISIBILITY VARCHAR2 (10) ,   
 SUBMITTED\_BY VARCHAR2 (200) ,   
 SUBMITTED\_ON TIMESTAMP DEFAULT systimestamp.  
 SHAPE VARCHAR2 (20)  
 )   
 TABLESPACE DATA   
 LOGGING   
;

**NOTE:** Columns such as FILE\_VISIBILITY, SUBMITTED\_BY, and SUBMITTED\_ON are for *demonstration purposes only*. They are not required.

An ORDS Endpoint has been created (with the below Resource Handler code). The following conditions exist:

* The endpoint expects multiple files and form data *in a JSON format* (i.e., the use of the :body\_json implicit parameter).
* The ORDS.BODY\_FILE\_COUNT function will be used to count the total files of the POST request.
* The ORDS.GET\_BODY\_FILE procedure will be used to store, in session, these files.

The following code example then performs an INSERT on the DEMO\_TABLE and relies upon various HTP procedures to “print” the results to a user, client, or application.

DECLARE  
 L\_PARAMETER\_NAME VARCHAR2(4000);  
 L\_FILE\_NAME VARCHAR2(4000);  
 L\_CONTENT\_TYPE VARCHAR2(200);  
 L\_FILE\_BODY BLOB;  
 L\_BODY\_JSON CLOB;  
BEGIN  
 L\_BODY\_JSON := :BODY\_JSON;  
 HTP.PARAGRAPH;  
 HTP.PRINT('Submitted By: ' || JSON\_VALUE(L\_BODY\_JSON, '$.submitted\_by'));  
 HTP.BR;  
 HTP.PARAGRAPH;  
 HTP.PRINT('File visibility status: ' || JSON\_VALUE(L\_BODY\_JSON, '$.file\_visibility'));  
 HTP.BR;  
 HTP.PARAGRAPH;  
 HTP.PRINT('Shape: ' || :SHAPE);  
 FOR I IN 1..ORDS.BODY\_FILE\_COUNT LOOP  
 ORDS.GET\_BODY\_FILE(  
 P\_FILE\_INDEX => I,  
 P\_PARAMETER\_NAME => L\_PARAMETER\_NAME,  
 P\_FILE\_NAME => L\_FILE\_NAME,  
 P\_CONTENT\_TYPE => L\_CONTENT\_TYPE,  
 P\_FILE\_BLOB => L\_FILE\_BODY  
 );  
 INSERT INTO DEMO\_TABLE (  
 FILE\_NAME,  
 FILE\_BODY,  
 CONTENT\_TYPE,  
 FILE\_VISIBILITY,  
 SUBMITTED\_BY,  
 SHAPE  
 ) VALUES ( L\_FILE\_NAME,  
 L\_FILE\_BODY,  
 L\_CONTENT\_TYPE,  
 JSON\_VALUE(L\_BODY\_JSON, '$.submitted\_by'),  
 JSON\_VALUE(L\_BODY\_JSON, '$.file\_visibility'),  
 :shape );  
 HTP.PARAGRAPH;  
 HTP.PRINT('Inserted File: ' || L\_FILE\_NAME);  
 HTP.BR;  
 END LOOP;  
END;

To test this :body\_json implicit parameter a curl command such as the one below may be used:

**NOTE:** You may have observed the included query parameter in the above POST request. In this example, we illustrate how automatic binding of query parameters (e.g., shape=triangle can be used in ORDS POST Resource Handlers).

curl --location 'https://gf641ea24ecc468-ordsdemo.adb.us-ashburn-1.oraclecloudapps.com/ords/ordsdemo/demo\_api/demo?shape=triangle' \  
--form 'files=@"demo-3.sql"' \  
--form 'files=@"demo-2.sql"' \  
--form 'submitted\_by="chris"' \  
--form 'file\_visibility="public"'

Accordingly, a client may respond with the following:

<p>  
Submitted By: chris  
<br />  
<p>  
File visibility status: public  
<br />  
<p>  
Shape: triangle  
<p>  
Inserted File: demo-3.sql  
<br />  
<p>  
Inserted File: demo-2.sql  
<br />

Along with an update to target the DEMO\_TABLE:

| ID | FILE\_NAME | FILE\_BODY | CONTENT\_TYPE | FILE\_VISIBILITY | SUBMITTED\_BY | SUBMITTED\_ON | SHAPE |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 144 | demo-2.sql | (BLOB) | application/x-sql | public | chris | 2024-11-06T15:00:46.494488Z | triangle |
| 145 | demo-3.sql | (BLOB) | application/x-sql | public | chris | 2024-11-06T15:00:46.49574Z | triangle |

1. In a scenario such as this, the form data in the POST body is formatted as a JSON object, and treated as a CLOB data type in the Oracle database. While *you can* store JSON in the Oracle database as JSON, VARCHAR2, CLOB, and BLOB, ORDS uses the CLOB data type, to ensure backward compatibility with earlier releases of the Oracle database. [↑](#footnote-ref-20)