Unable to Configure Return Properties on SmartObject

Background

A stored procedure in the Oracle RDBMS, which is comprised of input parameters and a reference cursor output, and which tests successfully, is to serve as the data layer object for a new K2 SmartObject.

In the course of building the SmartObject in K2 for Visual Studio, and despite having created properties intended for the return parameter, the service object method wizard prevents you from assigning output parameters to the service object method.

Discussion

When viewed from the SmartObject Service Tester, a stored procedure from within a package should look like this:[1]

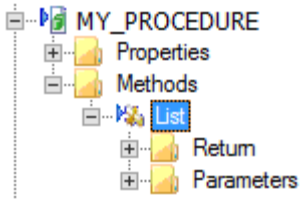

Figure 1

Beneath the List method, there should be two subfolders – one for parameters and one for the return value.

In the Service Object Method Wizard, you should see regions both for input and output parameters after adding the service object method (see *Figure 2*).

[1] When properly exposed to the K2 environment. If you can't find your package in SmartObject Services Tester, it's possible the proper permissions haven't been granted on the package object.

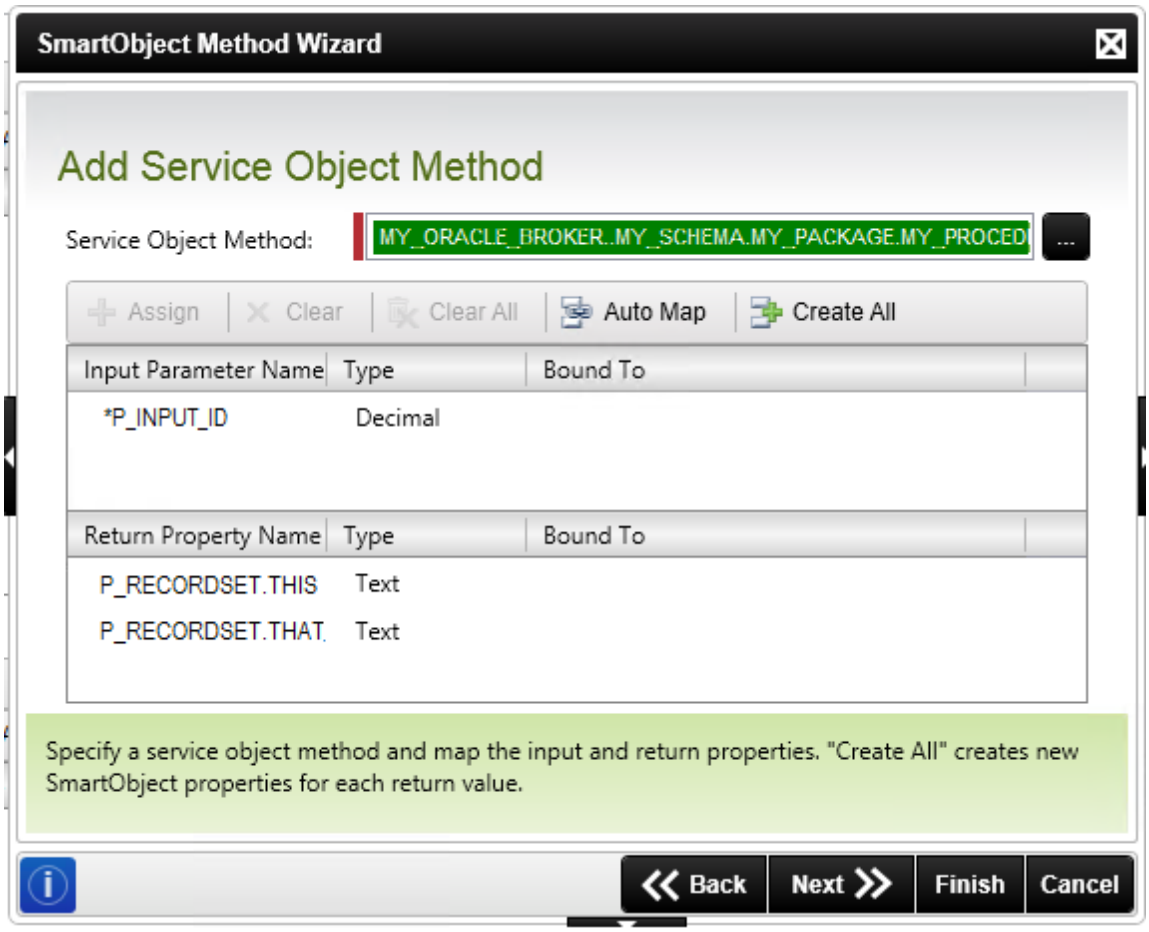

Figure 2

If your procedure resembles the one below, with no Return folder beneath the method name --

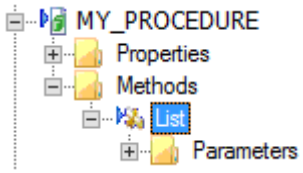

Figure 3

Then you may expect the absence of return parameter configuration in the SmartObject Method Wizard (see *Figure 4*):

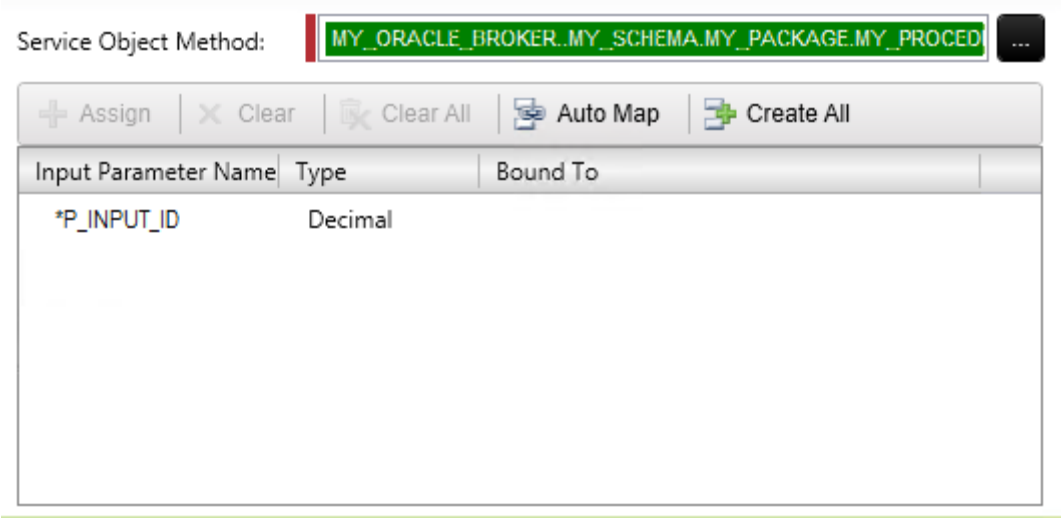

Figure 4

Remember, you can execute the procedure within Oracle SQL Developer and it will return the correct data. So what does K2 have against the cursor it's returning?

Solution

The head of the package correctly reports both the input and output parameters of your procedure, but K2 can't find the reference cursor you're returning.

Let's look at the procedure shown in *Figure 5*. This procedure grew from returning a single value into returning multiple values, and so a ref cursor was exchanged for the out parameter it was using.

The procedure is pretty easy to follow. A count is first done of the records in MY_TABLE (lines 10 through 13). If the data we're looking for isn't there, a count is executed against MY_OTHER_TABLE (lines 18 through 23). If the count is greater than zero, the reference cursor is engaged to return some values (lines 27 through 36).

The counts were a necessity under the original construction, because the original return value was being selected into the parameter variable, and the absence of a value would return an exception. So on the surface, it would seem that simply substituting the ref cursor for the previous parameter would make sense. And, the procedure executes correctly from within the Oracle SQL Developer software.

The trouble here is that K2 apparently cannot discern the state of the cursor from within the IF…END IF block (lines 27 through 36). The cursor isn't exposed anyplace else in the body but inside that block. For reasons that aren't clear to me yet, K2 simply cannot see the cursor (line 28).

```
1  PROCEDURE MY_PROCEDURE ( p_INPUT_ID IN MY_SCHEMA.MY_TABLE.ID%TYPE,
2                           p_RECORDSET OUT SYS_REFCURSOR) IS
3
4    v_MY_THINGS_COUNT NUMBER;
5
6    BEGIN
7
8        -- get a count
9
10       SELECT COUNT(T.MY_THINGS)
11       INTO v_MY_THINGS_COUNT
12       FROM  MY_SCHEMA.MY_TABLE T
13       WHERE T.ID = p_INPUT_ID;
14
15       -- if no things were returned, the record may have been moved.
16       -- try this:
17
18       IF v_MY_THINGS_COUNT = 0 THEN
19         SELECT COUNT(OT.MY_OTHER_THINGS)
20         INTO v_MY_THINGS_COUNT
21         FROM  MY_SCHEMA.MY_OTHER_TABLE OT
22         WHERE OT.ID = p_INPUT_ID;
23       END IF;
24
25       -- if more than zero records are found, return the data
26
27       IF v_MY_THINGS_COUNT > 0 THEN
28         OPEN p_RECORDSET FOR
29         SELECT THIS, THAT
30         FROM  MY_SCHEMA.MY_TABLE T
31         WHERE T.ID = p_INPUT_ID
32         UNION
33         SELECT OTHER_THIS AS THIS, OTHER_THAT AS THAT
34         FROM  MY_SCHEMA.MY_OTHER_TABLE OT
35         WHERE OT.ID = p_INPUT_ID;
36       END IF;
37
38   END MY_PROCEDURE;
```

Figure 5

To correct the condition, you must expose the ref cursor somewhere inside of the body and outside of the control block.

In the case of MY_PROCEDURE, the counts and control blocks may be removed altogether, because the desired data isn't being selected into a variable any longer. The end result is shown in *Figure 6*.

```
PROCEDURE MY_PROCEDURE ( p_INPUT_ID IN MY_SCHEMA.MY_TABLE.ID%TYPE,
                         p_RECORDSET OUT SYS_REFCURSOR) IS

  BEGIN

        OPEN p_RECORDSET FOR

        SELECT THIS, THAT
        FROM  MY_SCHEMA.MY_TABLE T
        WHERE T.ID = p_INPUT_ID
        UNION
        SELECT OTHER_THIS AS THIS, OTHER_THAT AS THAT
        FROM  MY_SCHEMA.MY_OTHER_TABLE OT
        WHERE OT.ID = p_INPUT_ID;

  END MY PROCEDURE;
```

Figure 6

Steps

(1) Modify the procedure to ensure your return parameter is not inside of a control block
(2) Compile the package
(3) Test the revised procedure to ensure it works
(4) Open SmartObject Services Tester
(5) Refresh the service instance
(6) Examine the procedure. You should now see the Return folder beneath the method, as pictured in *Figure 1*.
(7) Open K2 for Visual Studio and edit/create the SmartObject. You should be able to assign return properties, as pictured in *Figure 2*.