

ORACLE®

CPQ Cloud

CPQ Cloud - PCS Integration

IMPLEMENTATION GUIDE | CPQ 2016 R1

ORACLE®



Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.



ORACLE®



Contents

Disclaimer.....	1
Introduction.....	1
Purpose.....	1
Limitations.....	1
Scope.....	1
Implementation Overview.....	2
Implementation Prerequisites.....	3
CPQ Cloud Setup.....	3
Create a PCS Integration.....	3
Create a New Submit Action.....	5
Remote Approval BML Functions.....	6
Define Remote Approval BML Functions.....	7
CPQ REST Endpoints.....	8
PCS Setup Guidance.....	8
Web Form Parameters for Start Node and JSON.....	8
Service Activity Data Association to Call CPQ REST Endpoint.....	10
Appendix: BML Code Samples.....	11
Sample: JSON with attachments using urlmultipartbypost.....	11
Sample: JSON without attachments using urldatabypost.....	12
Sample: SOAP without attachments using urldatabypost.....	13
Sample: Approve/Reject BML.....	14
Sample: Revise BML.....	15



Introduction

The Oracle Configure, Price, and Quote (CPQ) Cloud application includes approval functionality and now provides customers with the ability to integrate CPQ Cloud with Oracle Process Cloud Service (PCS) or other remote approval systems to customize their approvals. Remote approval systems such as PCS also allow customers to consolidate all of their approvals into a central location and provide a common configuration and common attributes to manage approvals for an entire suite of applications. This eliminates the need to migrate approval data from one system to another. When customers choose remote approvals, CPQ sends approval requests to external approval systems for processing. Once complete, the associated approve or reject action is performed on the remote approval system, which then calls the appropriate approve or reject CPQ Cloud REST endpoint to update the CPQ quote history and status.

Purpose

The purpose of this CPQ Cloud – PCS Integration Implementation Guide is to provide an overview of remote approvals and instructions on how to implement the CPQ Cloud – PCS integration.

Limitations

Oracle has only tested the contents of this implementation guide on CPQ Cloud 2016 R1 and PCS 16.3.5 and above.

Note: Check with Oracle PCS customer support for the availability of this integration.

Scope

This implementation guide contains the following sections:

- Introduction – Provides a high-level overview of the benefits of integrating CPQ Cloud with PCS or other remote approval systems.
- Implementation Overview – Illustrates the remote approval call process and remote approval lifecycle.
- CPQ Cloud Setup - Provides CPQ Cloud setup instructions.
- CPQ REST Endpoints – Describes the PCS REST endpoints used by CPQ Cloud.
- PCS Setup Guidelines – Provides suggestions on how to setup PCS to integrate with CPQ Cloud

Note: Refer to the Appendix to view BML samples for remote approvals.

Implementation Overview

The following figure provides a visual depiction of the CPQ Cloud – PCS remote approval call process.

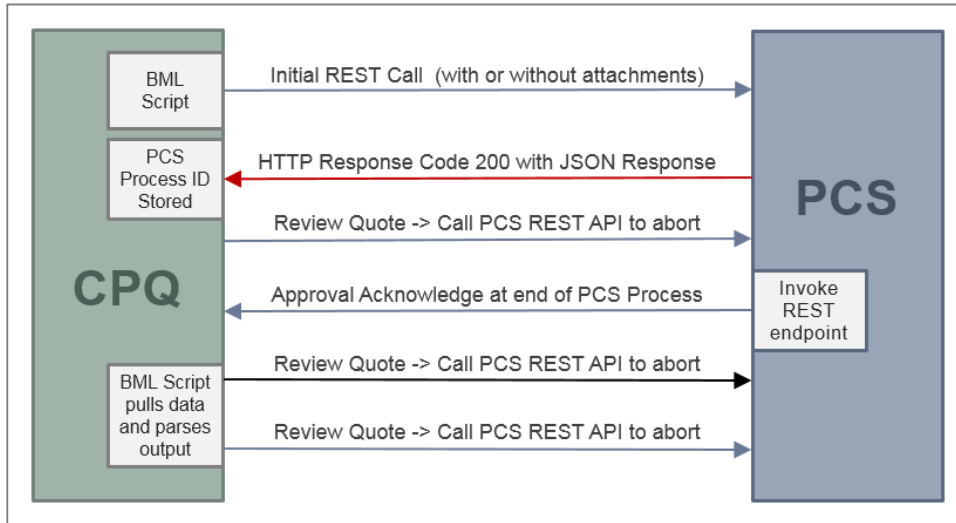


Figure 1: Remote Approval Call Process

The remote approval lifecycle is broken down into three steps (Submit, Remote Approval, and Final Approval), which are described below in Figure 2.

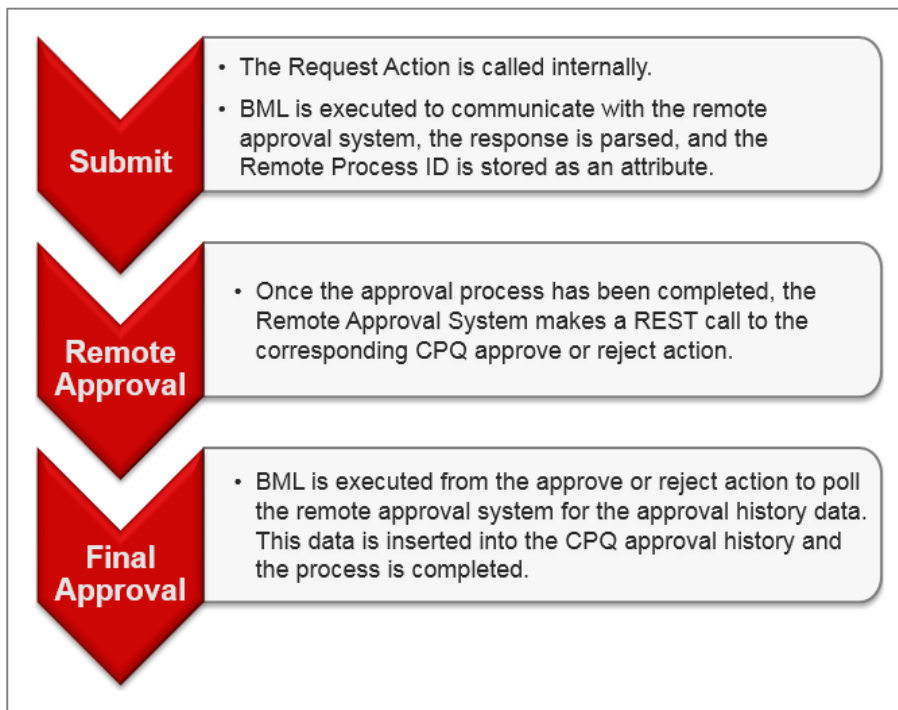


Figure 2: Remote Approval Lifecycle

Implementation Prerequisites

For instructions on how to setup a CPQ Cloud site, refer to the Administration Online Help for Oracle CPQ Cloud. You must login to CPQ Cloud as an administrator to view the Online Help.

In 2016 R1, CPQ Cloud introduces a new Submit action attribute to enable remote approvals. Use this new functionality to create a new Submit action for your remote approval system.

CPQ Cloud Setup

Before leveraging the pre-built PCS flows, administrators must establish a connection between PCS and CPQ Cloud.

Create a PCS Integration

Establish a connection from CPQ Cloud to PCS that complements the existing connection from PCS to CPQ Cloud.

1. In CPQ Cloud, go the **Admin** page.
2. Click **Integration Center** under **Integration Platform**.
The **Integration Center** page appears.
3. Click **Create Integration**.
The **Create Integration** page appears.

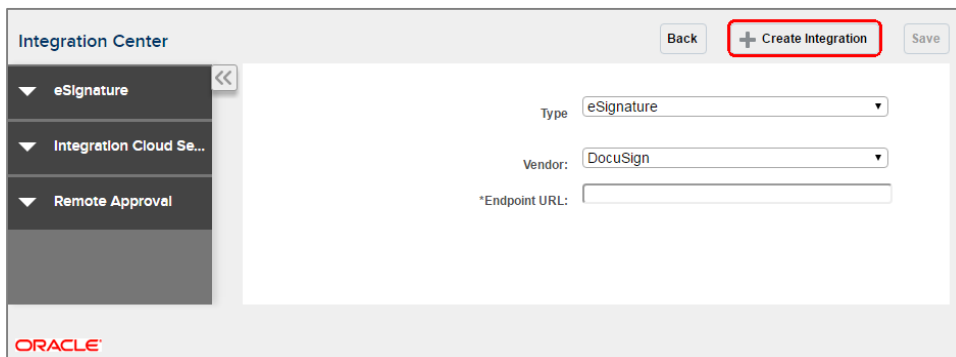


Figure 3: Create Integration

Note: The integration center only allows one integration for each integration type.

- If a Remote Approval integration exists, the administrator would modify the existing integration, or delete the existing integration before creating a new integration.
- If a user selects **Create Integration** and there are not any integrations available, the system displays an error message. Refer to Figure 4: No Integrations Available Error Message.

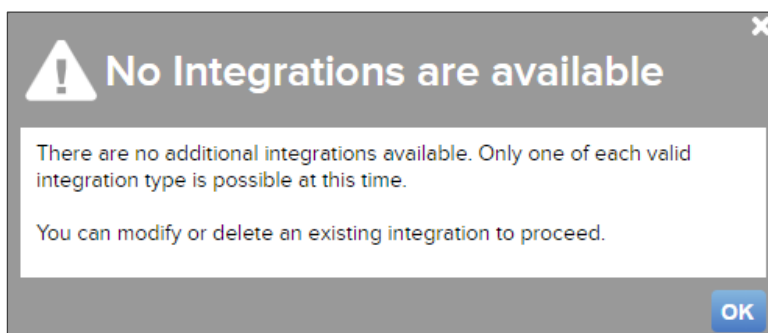


Figure 4: No Integrations Available Error Message

4. Select **Remote Approval** from the **Type** drop-down menu. There is a limit of one entry for each integration type. Any previously created integration types are disabled. Administrators can only select integration types that are not yet created. By default, the first valid option is selected.

Note: Changing the integration type immediately changes the available fields.

5. Enter a Name. Use a name that makes it easy to find your integration.

6. Enter the **Request URL** and **Revise URL**.

For example, Request URL and the Revise URL might look like the following:

```
https://<host:port>/bpm/api/1.0/processes
```

7. Enter your PCS **User name** and **Password**.

8. Click **Save** to enable your integration.

The screenshot displays the Oracle Integration Center interface for configuring a Remote Approval integration. The sidebar on the left shows a navigation menu with 'Remote Approval' selected. The main content area contains the following fields and controls:

- Type:** A dropdown menu set to 'Remote Approval'.
- *Name:** A text input field containing 'PCS'.
- Request URL:** A text input field containing 'https://slc04ny.oracle.com:7003/bpm/api/1.0/process'.
- Revise URL:** A text input field containing 'https://slc04ny.oracle.com:7003/bpm/api/1.0/process'.
- Username:** A text input field containing 'superuser'.
- Password:** A text input field with masked characters (dots).

At the top right of the form, there are three buttons: 'Back', '+ Create Integration', and 'Save' (which is highlighted with a red border).

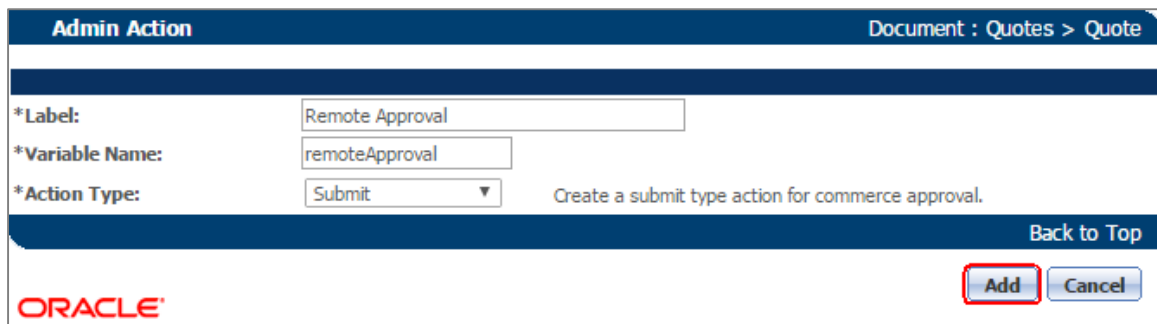
Figure 5: Remote Approval System Integration

Create a New Submit Action

Note: Administrators can also modify existing submit actions to enable remote approvals. Before changing the approval sequence to remote, close or finalize existing in process approvals.

Administrators must exercise discretion when changing the approval sequence selection.

1. Go to the **Admin** page.
2. Select **Process Definition** under **Commerce and Documents**.
The **Processes** page appears.
3. For the appropriate process, select **Documents** in the **Navigation** drop-down menu, then click **List**.
The **Document List** page appears.
4. For the appropriate document, select **Actions** in the **Navigation** drop-down menu, then click **List**.
The **Action List** page appears.
5. Select **Add** to create a new submit action.
The **Admin Action** page appears.
6. Enter a **Label** and a **Variable Name**.
7. Select **Submit** from the **Action Type** drop-down.
8. Click **Add** to create a new submit action.



The screenshot shows the 'Admin Action' page in Oracle. The page title is 'Admin Action' and the breadcrumb is 'Document : Quotes > Quote'. The form contains the following fields:

- *Label:** Remote Approval
- *Variable Name:** remoteApproval
- *Action Type:** Submit (dropdown menu)

Below the form, there is a note: 'Create a submit type action for commerce approval.' At the bottom right, there are two buttons: 'Add' (highlighted with a red box) and 'Cancel'. The Oracle logo is visible in the bottom left corner.

Figure 6: Create New Submit Action

9. From the Approval Sequence options, select **Use Remote** to enable the remote approval sequence.
10. Click **Update**.

The screenshot shows the 'Admin Action' configuration page for '(remoteApproval)'. The page has tabs for 'General', 'Modify', 'Integration', and 'Document Views'. The 'Approval Sequence' section is highlighted with a red box, showing two radio buttons: 'Use Approvals' (unselected) and 'Use Remote' (selected). Below this are buttons for 'Edit Approvals', 'Save and Edit Desktop Layout', and 'Save and Edit Mobile Layout'. There are also sections for 'Advanced Modify - Before Formulas', 'Advanced Modify - After Formulas', and 'Advanced Validation', each with radio buttons and a 'Define Function' button. At the bottom, there are buttons for 'Translations', 'Apply', 'Update' (highlighted with a red box), 'Update and New', and 'Back'. A 'Back to Top' link is also present.

Figure 7: Enable Remote Approval Sequence

Remote Approval BML Functions

Remote Approvals use BML functions for the following activities:

- Send a request to the remote approval system to initiate the approval process.
- Parse the response, return the process ID, and store the ID in the remote approval process ID attribute. The Submit action creates this attribute.
- Send the payload, with or without attachments, to PCS to start the PCS process.
- Use BMQL to fetch the username, password, and the URL fields defined in the integration center. Set the username and password in the header while using the URL functions.

Define Remote Approval BML Functions

Complete the following steps to define a request approval function:

1. Go to the **Admin** page.
2. Select **Process Definition** under **Commerce and Documents**.
The **Processes** page appears.
3. For the appropriate process, select **Documents** in the **Navigation** drop-down menu, then click **List**.
The **Document List** page appears.
4. For the appropriate document, select **Actions** in the **Navigation** drop-down menu, then click **List**.
The **Action List** page appears.
5. Set up the following sub-actions:
 - a. Request Approval – perform to Step 6 through Step 8.
 - b. Approve/Reject Function (*optional*) – perform to Step 9 through Step 11.
 - c. Revise Function (*optional*) – perform to Step 12 through Step 14.
6. Select remote approval **Request Approval** sub-action.
The remote approval **Request Approval Admin Action** page appears.
7. Click **Define Function** next to **Remote Call Processing**.



Figure 8: Define Function for Remote Call Processing

8. Enter and save the BML script for the request approval function. Refer to the following samples:
 - [JSON urlmultipartbypost](#) – Send attachments with the payload to start the PCS process.
 - [JSON urldatabypost](#) – Send only the JSON payload to start the PCS process.
 - [SOAP urldatabypost](#) – Send only the SOAP payload to start the PCS process.
9. Select the remote approval **Approve** sub-action.
The remote approval **Approve Admin Action** page appears.
10. Click **Define Function** next to **Remote Call Processing**.
11. Enter and save the BML script for the approve function.
For additional information, refer to the [Approve/Reject BML sample](#).
12. Select the remote approval **Revise** sub-action.
The remote approval **Revise Admin Action** page appears.
13. Click **Define Function** next to **Remote Call Processing**.
14. Enter and save the BML script for the revise function.
For additional information, refer to the [Revise BML Sample](#).

A new Submit action attribute called “remote_approval_process_id” is available for remote approvals. This attribute stores the value returned from the request approval BML function, which is typically processed for the PCS process instance. Use this attribute in the revise flow to cancel an existing PCS process instance.

Data 'Submit Attribute Set' for Document : Quote		
Name	Type	Description
My Approval@remoteSubmit	Text	This attribute lists all approval currently pending for current logged in user. This is not available as rule input.
Approval Status@remoteSubmit	Text Area	This attribute lists all currently pending active approvals for a transaction.
Approval History@remoteSubmit	Text Area	This attribute records approval history of a transaction. If a reason is reset, it is erased from the approval history.
Approval Revision@remoteSubmit	Integer	This attribute tracks the number of times a quote has been revised.
Remote Approval Process Id@remoteSubmit	Text	This attribute stores the corresponding process id in the remote approval system.

Figure 9: Remote Approval Process Id

CPQ REST Endpoints

Approve and reject sub actions are automatically exposed as REST endpoints. PCS uses these REST endpoints to notify CPQ when a quote is approved or rejected. When the REST endpoints are invoked with the “POST” operation, the corresponding BML in these actions is executed. The return value from the BML is used to update the approval history. If no BML is defined, then the approval history is not updated.

As shown below, the REST endpoints use the standard format:

```
http://hostname:port/rest/v1/commerceDocuments<processName>Quote/<bsid>/actions/  
<action_variable_name>
```

Note: When using remote approval functionality, do not use REST/SOAP web services to initiate approvals. The remote approval sequence does not support initiating approvals with REST/SOAP web services. The only web service supported, to notify CPQ Cloud of the approval status, is on REST Endpoints for approve and reject sub actions.

PCS Setup Guidance

The PCS setup information provided in this section is intended as guidance only. With future versions of PCS, some portions of the user interface may change. For additional setup guidance, refer to the PCS documentation or contact PCS support.

Web Form Parameters for Start Node and JSON

The JSON string defined in the BML Request Approval sub action should match the Web Form parameters for the PCS process Start Node. If they do not match, an error displays when executing the request approval action.

If the Start node in PCS is defined to accept a structure like header and lines, then the JSON in CPQ Cloud should contain the same XML tags, header, and line data.

Figure 10: Web Form Parameters for Start Node

JSON string in BML Request Approval sub action:

```
json = "{\"processDefId\":\"default~CPQApprovalDemo!3.0~QuoteApprovalProcess\",
  \"serviceName\":\"QuoteApprovalProcess.service\", \"operation\":\"start\", \"payload\":
  \":<quot:start xmlns:quot='http://xmlns.oracle.com/bpmn/bpmnCloudProcess/CPQApprovalDemo/Quo
  <requestor>Suyog</requestor>
  <quoteId>" +bs_id + "</quoteId>
  <quoteTotal>12567</quoteTotal>
  <discountPercentage>23</discountPercentage>
  <requestorEmail>approve_pcssubmit</requestorEmail>
  </quot:start\""}, I
  \"action\":\"submit\"}";
```

Service Activity Data Association to Call CPQ REST Endpoint

PCS uses a service activity to call the CPQ REST endpoint for the approve and reject action. In the data association for this service activity, verify the `bs_id` and sub action variable name when constructing the URL to invoke.

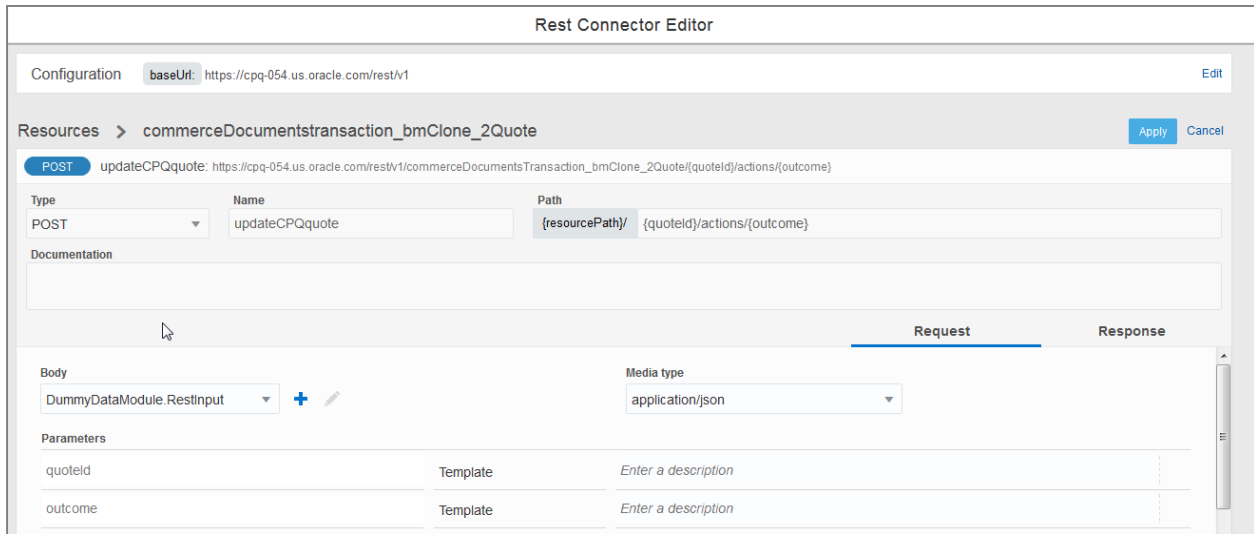


Figure 11: Rest Connector Editor

An example data association for a service activity is shown below in Figure 12: Data Association.

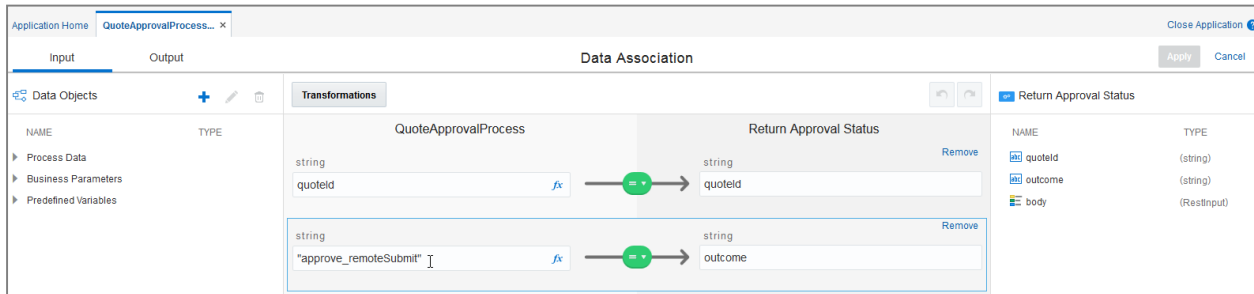


Figure 12: Data Association

Note: In SSL enabled environments, the CPQ administrator should work with the PCS administrator to ensure the CPQ server is trusted by the PCS domain. This setup is not usually required in **PODs** where the applications are typically all on the same domain. An SSL path error typically displays at the end of the process during the CPQ REST call from PCS. When the error displays, this setup procedure is needed. For additional information on the setup, contact PCS Cloud Support.

Appendix: BML Code Samples

Sample: JSON with attachments using urlmultipartbypost

The attributes used -

```
bs_id          String .....                               bs_id
pcsAttachment  String (Attachment)      pcsAttachment - File attachment attribute
+++++
records = bmql("select username, password, reviseUrl, requestUrl from
integration.remoteApproval");
username=""; password=""; reviseUrl=""; requestUrl="";
for record in records {
    reviseUrl=get(record, "reviseUrl");
    requesturl=get(record, "requestUrl");
    username = get(record, "username");
    password = get(record, "password");
}
// This JSON is provided as an example. This needs to be changed according to the input
// defined for the PCS process. The CPQ Admin should work with PCS Admin to get details
on
// what input is required to be passed into the process, based upon what has been
defined.
json = "{\"processDefId\":\"default~CPQApprovalDemo!3.0~QuoteApprovalProcess\",
\"serviceName\":\"QuoteApprovalProcess.service\", \"operation\":\"start\", \"payload\":
\"<quot:start
xmlns:quot='http://xmlns.oracle.com/bpmn/bpmnCloudProcess/CPQApprovalDemo
/QuoteApprovalProcess'><requestor>Suyog</requestor><quoteId>\" +bs_id + "</quoteId>
<quoteTotal>12567</quoteTotal><discountPercentage>23</discountPercentage>
<requestorEmail>approve_pCSsubmit</requestorEmail></quot:start>\", \"action\":\"Submit\"
}";
attachmentsDict = dict("dict<anytype>");
attachment1 = getattachmentdata(pcsAttachment, true);
put(attachmentsDict, "attachment1", attachment1);
header = dict("string");
put(header, "Accept", "application/json");
encodeCredential = encodebase64(username+":"+password);
put(header, "Authorization", "Basic "+encodeCredential);
//response = urlmultipartbypost(requestUrl,json,header); In case, there are no
attachments
response = urlmultipartbypost(requesturl,json,header,attachmentsDict);
if(get(response, "Status-Code") == string(200)) {
    processString= json(get(response, "Message-Body"));
    return jsonget(processString, "processId", "String");
}
else {
    throwerror(get(response, "Message-Body"));
    return "";
}
}
```

Sample: JSON without attachments using urldata bypost

The attributes used -

bs_id String

bs_id

```
+++++
records = bmql("select username, password, reviseUrl, requestUrl from
integration.remoteApproval");
username=""; password=""; reviseUrl=""; requestUrl="";
for record in records {
    reviseUrl=get(record, "reviseUrl");
    requesturl=get(record, "requestUrl");
    username = get(record, "username");
    password = get(record, "password");
}
json = "{\"processDefId\":\"default~CPQApprovalDemo!3.0~QuoteApprovalProcess\",
\"serviceName\":\"QuoteApprovalProcess.service\", \"operation\":\"start\", \"payload\":
\"<quot:start
xmlns:quot='http://xmlns.oracle.com/bpmn/bpmnCloudProcess/CPQApprovalDemo
/QuoteApprovalProcess'><requestor>Suyog</requestor><quoteId>\" +bs_id + \"</quoteId>
<quoteTotal>12567</quoteTotal><discountPercentage>23</discountPercentage>
<requestorEmail>approve_pCSsubmit</requestorEmail></quot:start>\", \"action\":\"Submit\"
}\"";
header = dict("string");
put(header, "Accept", "application/json");
encodeCredential = encodebase64(username+":"+password);
print(username);
print(password);
put(header, "Authorization", "Basic "+encodeCredential);
put(header, "Content-Type", "application/json");
response = urldata bypost(requestUrl, json, "", header, true);
//response = urlmultipart bypost(requestUrl, json, header);
//processString= get(response, "Message-Body");
if(find(lower(response), "error")<0) {
    return jsonget(json(response), "processId", "String");
} else {
    throwerror(response);
    return "";
}
}
```

Sample: SOAP without attachments using urldatabypost

The attributes used -

```
bs_id      String
bs_id
+++++
records = bmql("select username, password, reviseUrl, requestUrl from
integration.remoteApproval");
username=""; password=""; reviseUrl=""; requestUrl="";
for record in records {
    reviseUrl=get(record, "reviseUrl");
    requesturl=get(record, "requestUrl");
    username = get(record, "username");
    password = get(record, "password");
}
xml="<soap:Envelope xmlns:soap=\"http://www.w3.org/2003/05/soap-envelope\"><soap:Body>
<ns1:start xmlns:ns1=\"http://xmlns.oracle.com/bpmn/bpmnCloudProcess/CPQApprovalDemo
/QuoteApprovalProcess\"><requestor>suyog</requestor><quoteId>+bs_id+</quoteId>
<quoteTotal>10000</quoteTotal><discountPercentage>23</discountPercentage>
<requestorEmail>skulkarn</requestorEmail></ns1:start></soap:Body></soap:Envelope>";
header = dict("string");
put(header, "Content-Type", "application/soap+xml");
response = urldatabypost(requestUrl, xml, "",header,true);
if(find(lower(response), "error")<0) {
    return jsonget(json(response), "processId","String");
} else {
    throwerror(response);
    return "";
}
```


Sample: Approve/Reject BML

The attributes used -

```
_quote_remote_approval_process_id_remoteSubmit ..... String
RemoteProcessId
+++++
records = bmql("select username, password, reviseUrl, requestUrl from
integration.remoteApproval");
username=""; password=""; reviseUrl=""; requestUrl="";
for record in records {
    reviseUrl=get(record, "reviseUrl");
    requesturl=get(record, "requestUrl");
    username = get(record, "username");
    password = get(record, "password");
}
returnString="";
PullURL = requestUrl+"/"+ _quote_remote_approval_process_id_remoteSubmit +
"/audit?graphic=N";
header = dict("string");
encodeCredential = encodebase64(username+": "+password);
put(header, "Authorization", "Basic "+encodeCredential);
response = urldata(PullURL, "get", header);
returnValue = json(get(response, "Message-Body"));
print("return Value - ");
print(returnValue);
historyValues = jsonpathgetsingle(returnValue , "$.processHistory[2].taskHistory",
"jsonarray");
print(historyValues);
strArray= String[jsonarraysize(historyValues)];
i=0;
for s in strArray {
    jsonDict= jsonarrayget(historyValues, i, "json");
    actionName = jsonget(jsonDict, "actionName");
    comment=jsonget(jsonDict, "reason");
    if(actionName=="Task Completed - Approved") {
        actionName=BM_REMOTE_APPROVAL_STATUS_APPROVED;
    } elif(actionName=="Task Completed - Rejected"){
        actionName=BM_REMOTE_APPROVAL_STATUS_REJECTED;
    } else {
        ..... if(comment=="") {
            comment=actionName;
        }
        else {
            comment=actionName+"-"+comment;
        }
        actionName=BM_REMOTE_APPROVAL_STATUS_CUSTOM;
    }
    datel = jsonget(jsonDict, "updatedDate");
    returnString=returnString+jsonget(jsonDict, "displayName")+"~"+datetostr
(strtojavadata(datel, "yyyy-MM-dd HH:mm:ss"), "MM/dd/yyyy hh:mm
a")+"~"+actionName+ "~" + comment + "~||";
    i=i+1;
}
return returnString;
```

Sample: Revise BML

The attributes used -

```
_quote_remote_approval_process_id_remoteSubmit ..... String
  RemoteProcessId
+++++
records = bmql("select username, password, reviseUrl, requestUrl from
integration.remoteApproval");
username=""; password=""; reviseUrl=""; requestUrl="";
for record in records {
    reviseUrl=get(record, "reviseUrl");
    requesturl=get(record, "requestUrl");
    username = get(record, "username");
    password = get(record, "password");
}
header = dict("string");
encodeCredential = encodebase64(username+":"+password);
put(header, "Authorization", "Basic "+encodeCredential);
put(header, "Content-Type", "application/json");
cancelBody = "{\"id\": \"cancel\"}";
url = reviseUrl+"/"+ _quote_remote_approval_process_id_remoteSubmit;
response = urldata(url, "PUT", header, cancelBody);
returnValue = get(response, "Message-Body");
if(get(response, "Status-Code") == string(200)) {
    return "-1";
}
else {
    throwerror(get(response, "Message-Body"));
    return "";
}
```



Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA


Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

 blogs.oracle.com/oracle

 facebook.com/oracle

 twitter.com/oracle

 oracle.com

Integrated Cloud Application & Platform Services

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0116