

Advanced Collections - Creating Tasks for 'Pre-Delinquency Calls' using JTF_TASKS_PUB API

Author: Anil Patil

Created: May 4, 2007

Product: Advanced Collections

Version: 11.5.10 RUP3.1

Objective

The Objective of this document is to explain how we created tasks for 'Pre-Delinquency Calls' and assigned the tasks to the Collectors.

Business Requirement

One of our Clients has implemented Advanced Collections and is running their strategies at 'Account Level'. To manage the collections activities proactively, they felt the need of notifying the Collectors in advance of some high dollar amount invoices that would get due in the next 'x' number of days. The Collectors can then give a pre-delinquency phone call to the customer to check if all is fine and if they can expect the payment on or before the due date. This would help them in better Collections. If there are any issues like Invoice not received, Error in Billing or any other issue, they can get it resolved well in advance. The need was for a fully automated solution that would create a task to notify the collectors and the collectors can access and work on their task from their Collections Work Queue.

Benefits

1. This eliminates the need for the Collections Manager to run AR reports and notify the Collectors by email/manually of the high dollar amount invoices that would be due.
2. Tasks are automatically created and assigned to the Collector of the Account.
3. The history of the 'Pre-Delinquency Call' tasks and its notes is maintained in the application.
4. Integrated Solution with Advanced Collections Module. The collectors don't have to navigate outside the Advanced Collections Applications to work on these tasks.

Definitions

1. Pre-Delinquent Invoice – Invoices that are not yet past due.
2. Delinquent Invoice – Invoices that are past due.
3. High Dollar Amount Invoices – Invoices that are above the high value threshold as defined by the Collections Manager
4. Pre-Delinquent Tasks – Tasks that gets automatically created by the system. These tasks are assigned to the collector. This task reminds the Collector to make a Pre-Delinquent Phone Call to the Customer to check if everything is OK to make the payment by the Due Date.

Assumptions

1. The Collections Work Queue is configured to display the Task node for the Collectors.

Solution

This business requirement could not be implemented using the Strategy Work Items functionality, because the strategy is configured to run at the account level and the strategy gets assigned to the Account only when the Account Status is Delinquent. The need here was to create a task per invoice even before the invoice gets delinquent. The solution, we implemented for this was to create a task programmatically using the JTF_TASKS_PUB.CREATE_TASK API and assigning the task to the Collector.

A custom concurrent program was written and this program was scheduled to run daily. The parameters for this program were

1. High Dollar Amount: Only those invoices having the invoice amount more than the 'High Dollar Amount' parameter gets selected for the pre-delinquency calls.
2. Number of Days: This parameter signifies how many days prior to the Invoice due date, the 'Pre-Delinquency Call' task gets created.

Example for Invoice selection and task creation dates.

The parameter values are as follows

High Dollar Amount: 50000

Number of Days: 10

Invoice Number	Invoice Amount	Invoice Due Date	Invoice selected for Pre-Delinquency Call	Task Creation Date
123	25000	15-May-2007	No	-
456	75000	18-May-2007	Yes	8-May-2007
789	35000	25-May-2007	No	-
987	90000	27-May-2007	Yes	17-May-2007

Business Rules

1. If a task has been created against one pre-delinquent high-value invoice, the task should not be created again for the same invoice.
2. The Collections Manager will specify the 'High Dollar Amount' and the 'Number of Days'.
3. One task gets generated for each high dollar amount Invoice. The invoice number is shown in the task description.
4. The task gets assigned to the Collector of the Account. If there is no valid Collector, then it gets assigned to the default strategy resource as setup in the profile options.

High Level Algorithm for the Concurrent Program

1. Select the transactions for which the task needs to be created.
2. Check if the pre-delinquent task has already been created for the transaction.
3. If 'YES', return to End. Else go to '4'
4. Provide the source data as input parameters to the JTF_TASKS_PUB.CREATE_TASK API
5. Capture the return status flag from the API
6. Provide the output of the Tasks Created in the Output file.

Sample Code

1. Query to select the transactions

```
SELECT payment_schedule_id
       , trx_number
       , customer_id
       , amount_due_remaining
       , trx_date
       , due_date
       , acctd_amount_due_remaining
FROM ar_payment_schedules ps
WHERE ps.CLASS IN ('INV', 'DM', 'CB')
      AND ps.gl_date <= SYSDATE
      AND ps.gl_date_closed > SYSDATE
      AND ps.status = 'OP'
      AND ps.amount_due_remaining > &high_dollar_amount
      AND ps.due_date <= SYSDATE + &number_days
      AND SYSDATE <= ps.due_date;
```

2. Query to check if the task is already created

```
SELECT count(*)
INTO x_count
FROM jtf_tasks_b
WHERE cust_account_id = invrec.customer_id
      AND scheduled_start_date >= SYSDATE - &number_days
      AND attribute1 = invrec.payment_schedule_id ;
```

In the above query, invrec is a record_type consisting the source data. If the x_count variable is not zero, it means that the task is already created. The payment_schedule_id of the trx for which the task is created is stored in attribute1 column of jtf_tasks_b.

3. API Code

```
DECLARE
  l_return_status    VARCHAR2 (1);
  l_msg_count        NUMBER;
  l_msg_data         VARCHAR2 (1000);
  l_task_id          NUMBER;

BEGIN
```

```

jtf_tasks_pub.create_task
(p_api_version           => 1,
 p_init_msg_list        => 'T',
 p_commit               => 'T',
 p_task_name            => 'Pre-Delinquency Call Reminder',
 p_task_type_name       => 'Callback',
 p_description           => 'Invoice: ' || invrec.trx_number ,
 p_task_status_name     => 'Assigned',
 p_task_priority_name   => 'Medium',
 p_owner_type_code      => 'RS_EMPLOYEE',
 p_owner_id             => v_resource_id,
 p_customer_id          => v_party_id,
 p_cust_account_id      => invrec.customer_id,
 p_scheduled_start_date => SYSDATE,
 p_source_object_type_code => 'IEX_ACCOUNT',
 p_source_object_id     => invrec.customer_id,
 p_attributel           => invrec.payment_schedule_id,
 x_return_status        => l_return_status,
 x_msg_count            => l_msg_count,
 x_msg_data             => l_msg_data,
 x_task_id              => l_task_id
);

IF x_return_status = 'S'
  THEN
    fnd_file.put_line (fnd_file.output,
                      invrec.trx_number
                      || ' '
                      || invrec.amount_due_remaining
                      || ' '
                      || invrec.trx_date
                      || ' '
                      || invrec.due_date
                      );
  ELSE
    fnd_file.put_line (fnd_file.output,
                      '** ERROR IN CREATING TASK FOR INVOICE : '
                      || invrec.trx_number
                      || ' **'
                      );
END IF;

END;

```

In the above code

- invrec is a record_type consisting the source data.
- The variable v_resource_id is the resource_id of the collector of the account and can be retrieved using the following query

```

SELECT resource_id
INTO v_resource_id
FROM ar_collectors

```

```
WHERE collector_id = (SELECT collector_id
                      FROM hz_customer_profiles
                      WHERE cust_account_id = invrec.customer_id
                      AND status = 'A'
                      AND site_use_id IS NULL) ;
```

If there is no collector on the account or if the collector does not have a resource_id, we use the default strategy resource

```
V_resource_id := fnd_profile.VALUE ('IEX_STRY_DEFAULT_RESOURCE');
```

- The variable v_party_id is the party id of the account and can be retrieved using the following query

```
SELECT party_id
FROM hz_cust_accounts
WHERE cust_account_id = invrec.customer_id;
```

Summary

This document details one approach for creating 'Pre-Delinquency Call' Tasks and assigning the task to the collector of the account.

References

Oracle Common Applications Components – API Reference Guide
Oracle Advanced Collections Implementation Guide
Oracle Advanced Collections User Guide