Upsert Job Queue for SugarCRM

Technical Design

Prerequisites

System requirements for installation

Supported Versions

- 11.x
- 12.x

Supported Editions

- Sell
- Serve
- Enterprise
- Professional

Supported Databases

- MySQL
- MSSQL
- Oracle
- DB2

Supported Languages

- English

Overview

The Job Queue is designed to be a processing framework that handles the following:

- Dependencies
- Retries
- Logging
- Completion/Failure Alerts

By handling these features, administrative and developer users have a toolset to quickly diagnose and remedy issues resulting from integrations to Sugar.

Job Queue API

The Job Queue module can accept jobs by directly sending a POST request to the module's create-job or create-jobs endpoint.

POST /rest/v11/upsert_CustomQueue/create-job

```
{
    'job_module': 'Quotes',
    'job': 'Import',
    'context': {
        'name': 'My Quote'',
        'purchase_order_num': '12345',
        'my_custom_field_c': 'Custom Value'
    },
    'options': {
    }
}
```

POST /rest/v11/upsert_CustomQueue/create-jobs

```
{
    'jobs': [
        {
            'job_module': 'Quotes',
            'job': 'Import',
            'context': {
```

```
            'name': 'My Test Quote',
            'purchase_order_num': '12345',
            'my_custom_field_c': 'Custom Value'
        },
        'options': {
        }
    }
  ]
}
```

These endpoints will automatically place a record in the queue to be cycled by the Sugar cron. Once picked up by the cron, it will find the code we specify for the Quotes / Import job:

```
/**
* Imports a quote record
*
* @param mixed $job
*
* @return int
*/
public static function importQuote($job)
{
  //get the context
  $fields = $job->getContext();

  //log our fields
  $job->addLog('Importing quote with fields:', $fields);

  if (!isset($fields['purchase_order_num'])) {
     $job->addLog('Purchase order not found in payload');
     return static::$CODE_FAILED;
  }

  //see if we have an existing quote
  $sq = new \SugarQuery();
  $sq->from(\BeanFactory::newBean('Quotes'));
  $sq->select(['id']);
  $sq->where()
     ->in('purchase_order_num', [
        $fields['purchase_order_num'],
     ]);
  $sq->orderBy('date_modified', 'ASC');
```

```
    $sq->limit(1);

    $id = $sq->getOne();

    if (!empty($id)) {
        $message = "Updating existing record";
        $quote = \BeanFactory::getBean('Quotes', $id);
    } else {
        $message = "Creating new record";
        $quote = \BeanFactory::newBean('Quotes');
    }

    //create/update the record
    $quote->fromArray($fields);

    $changes =
\Sugarcrm\Sugarcrm\custom\Upsert\Custom\Libraries\Sugar\Classes\Helpers\Bean::getDataCh
anges($quote);

    if ($changes) {
        $job->addLog($message, $changes);
        $quote->save(false);
    } else {
        $job->addLog("No changes made to existing record. ({$id})");
    }

    //set the quote as the job parent
    $job->parent_type = $quote->module_name;
    $job->parent_id = $quote->id;

    return static::$CODE_COMPLETED;
}
```

Based on the code above, the quote record will be created or updated with a log of its actions and marked as completed. In a scenario where we only wanted our quotes to be updated if they exist, our function could look like this:

```
  /**
* Imports a quote record
*
* @param mixed $job
*
```

```
* @return int
*/
public static function importQuote($job)
{
   //get the context
   $fields = $job->getContext();

   //log our fields
   $job->addLog('Importing quote with fields:', $fields);

   if (!isset($fields['purchase_order_num'])) {
      $job->addLog('Purchase order not found in payload');
      return static::$CODE_FAILED;
   }

   //see if we have an existing quote
   $sq = new \SugarQuery();
   $sq->from(\BeanFactory::newBean('Quotes'));
   $sq->select(['id']);
   $sq->where()
      ->in('purchase_order_num', [
         $fields['purchase_order_num'],
      ]);
   $sq->orderBy('date_modified', 'ASC');
   $sq->limit(1);

   $id = $sq->getOne();

   if (!empty($id)) {
      $message = "Updating existing record";
      $quote = \BeanFactory::getBean('Quotes', $id);
   } else {
      $job->addLog('Existing record not found');
      return static::$CODE_FAILED;
   }

   //create/update the record
   $quote->fromArray($fields);

   $changes =
\Sugarcrm\Sugarcrm\custom\Upsert\Custom\Libraries\Sugar\Classes\Helpers\Bean::getDataCh
anges($quote);

   if ($changes) {
```

```
        $job->addLog($message, $changes);
        $quote->save(false);
    } else {
        $job->addLog("No changes made to existing record. ({$id})");
    }

    //set the quote as the job parent
    $job->parent_type = $quote->module_name;
    $job->parent_id = $quote->id;

    return static::$CODE_COMPLETED;
}
```

This will return a failure status for the job to be reviewed by an administrator and show in dashlets and reports.

Custom Job Queue APIs

```
    /**
     * Custom endpoint function to create multiple jobs for importing quotes
     *
     * @param \ServiceBase $api
     * @param array $args
     * @return array
     */
    public function customEndpoint(\ServiceBase $api, array $args)
    {
        $jobIds = [];
        foreach($args['quotes'] as $quoteFields) {
            $job = \upsert_CustomQueue::create('Quotes', 'updateQuote', $quoteFields,
['prevent_duplicates' => true]);
            $jobIds[] = $job ? $job->id : false;
        }

        return $jobIds;
    }
```

Job Options

Jobs have options that can be passed to allow for things like duplicate preventions, failure alerts, and dependencies;

prevent_duplicates	boolean	Prevents duplicate duplicates from being created with the same job and context.
retries_allowed	integer	The number of time a job can be retried before resulting in failure.
dependent_on_id	string	Specifies the id of a previous job that must be completed before the current job can be executed.
assigned_user_id	string	Assigns the job record to a specific user.
alert_on_failure	boolean	Whether or not to notify the assigned user upon job completion.
alert_on_completion	boolean	Whether or not to notify the assigned user upon job completion.
execute_immediately	boolean	Whether we should execute the job immediately or allow the cron to pick up the job.

Failure Alerts

When a job has failed, the system can alert the assigned user of the record with an in-app notification and/or email. The user can enable notification preferences in their Sugar profile.