Workflows Progress Design

* 1. **Background and Requirement**

Some of RackHD tasks require long time to execute such as OS bootstrap, security erase. But there is no progress notification mechanism existed for these long run tasks, so that user can only actively poll task graph to get a little status information. We need to provide some notification mechanism to expose more internal status and progress details for long-run task so as to get better user experience.

Required features for progress design include:

* 1. Provide workflow level progress indication mechanism
	2. Provide task level progress indication mechanism for long run tasks as:
		+ OS installation
		+ Secure erase
	3. Report progress as % complete
	4. User should be able to retrieve progress information in:
		+ AMQP channel
		+ User provided webhook

More requirement details can be found in below link:

<https://rackhd.atlassian.net/browse/RI-46>

**2. Progress Feature design**

**2.1 Existing progress information and limitation**

Actually RackHD did provide 4 kinds of basic progress events through AMQP channel when it runs any workflow :

* + Workflow started events, workflow finished events, task started events, tasks finished events

However there are some problems with those existing events to measure workflow progress:

* + Each type of message has independent payload format and different routing keys. It is difficult for customer subscribe dedicated progress message.
	+ They are not organized in a common progress scope. User can receive those events in sequence and know what have been done. But it is difficult to know what remains and can't measure a progress and get progress completion ratio.
	+ There is no task internal progress information. For a RackHD long-run workflows, if we have no idea what is going on inside the task, user can not receive any message for a very long time till tasks being completed.

Workflow progress design is going to solve those 3 problems and provide a direct progress to user.

**2.2 Progress message design**

**2.2.1 Progress message format**

Progress information will be described in an object with 4 attributes:

|  |  |  |
| --- | --- | --- |
| **properties** | **Type** | **Description** |
| maximum | Integer | Maximum step quantity for a workflow or a task. For tasks with continuous progress, it is 100. |
| value | Integer | Completed step quantity for a workflow or a task. It is an integer between 0 and maximum.For tasks with continuous progress, it varies from 0-100, which is inversely calculated from percentage and rounded to integer if calculation gives non-integer value. |
| percentage | String | Completion ratio of a executing workflow or task. Normally *value* divided by *maximum* will give *percentage*. However in the case that tasks have continuous progress and we can only get percentage instead of *maximum/value*, *maximum* will be always set to 100 and *value* will be set to the percent number. For example, a percentage “65%” will give *maximum* 100 and *value* 65. |
| description | String | Short description for events that publish progress messages. In the case value varies very slow, a varying description can also help to understand of a task is dead or alive. |

Below is an example of progress information payload for a workflow that has 4 tasks and we have just finished the first task. Percentage is 25% given by 1 / 4.

progress: {
 value: 1,
 maximum: 4,
 description: 'Task "Install CentOS" started',
 percentage: '25%'
}

Below is a complete RackHD progress message payload:

{
 progress: {
 value: 1,
 maximum: 4,
 description: 'Task "Install CentOS" started',
 percentage: '25%'
 },
 graphName: 'Install CentOS',
 graphId: '12a8f275-7abf-46ee-834b-6aa34cce8d78',
 nodeId: '58542c752be86d0672cef383',
 taskProgress: {
 taskId: 'cb7d5793-abcf-4a7f-aef6-e768e999de1d',
 taskName: 'Install CentOS',
 progress: {
 value: 0,
 maximum: 4,
 description: 'Task started',
 percentage: '0%'
 }
 }
}

A complete progress message contains two levels of progress information as well as some useful items like graphId, graphName, which can be used to filter and group progress messages.

RackHD "publisProgressEvent" is a common interface to publish progress messages. Among all data above, progress value, maximum description, graphId and taskId must provided by developer. Other items will be filled automatically by the progress publish method if not provided.

**2.2.2 Progress message user interface**

Progress message will be published in AMQP channel and posted to any web hook URL if it meets hook filtering condition. Common RackHD event headers will be added during publish action.

Here is some basic information to subscribe progress message via AMQP channel:

* + Exchange:  on.events
	+ Routing Key: graph.progress.updated.information.<graphId>.<nodeId>

For more details on RackHD events header, AMQP or web hook design, please refer to:

<http://rackhd.readthedocs.io/en/latest/rackhd/event_notification.html>

**2.3 Progress measurement design**

**2.3.1 Events based measurements**

There are two ways to measure workflow progress:

* + Time-based: progress is measured based on time consuming. Progress information indicates time consumed by an active workflow and can be used to estimated remaining executing time
	+ Events-based: progress is measured based on import events. Progress information indicates how many events happened and how many events remains.

It is difficult for RackHD to estimated how much time a running workflow will consume. For example, an OS installtion workflow time consuption will be impacted by network bandwidth, a reboot task time consumption varies with different hardware platforms.

Thus RackHD progress is events-based. And even though RackHD provides percentage number as progress measurement, RackHD progress percentage is not always proper to be used for workflow executing time estimation.

**2.3.2 Progress events for long run tasks**

 To measure long run task progress, two types of new progress events are designed:

* **Milestone events**
In some cases RackHD can’t easily get progress information direct ly, some milestones are created to divide a task into several small sections. Progress message will be sent if any of those milestones is achieved. Progress is easily measured in steps in this case.
* **Progress timer timeout events.**In some cases progress of a task is continuous, RackHD can’t easily measure steps of task and should avoid keeping sending progress information all the time. Thus a repeating timer is set and progress information is publish when it timeouts. Also in cases progress is updating rather slow and progress numbers are not even updating within a long duration, timeout events is a mechanism to indicates tasks is still running instead of hung or dead. Progress is measured with percentage in this case.

Section 2.5 shows how milestone events is used in OS installation and section 2.5 shows progress timer timeout events.

**2.3.3 Two levels of progress measurement**

RackHD progress information contains two levels of progress as show in section 2.2.1:

* + ***Task level progress*:** progress measurement of the executing task of an active workflow.

A non-long-run task will complete in short time and only the started and finished events can be sensed. Thus only two task progress messages will be published for tasks started and task finished. Progress will be measure with percentage and only 0% and 100% will be available.

Section 2.4 and 2.5 describe two typical long run task progress design, task progress is measured on milestone events and progress timer timeout events

* + ***Workflow level progress*:** progress measurement of an active workflow.

Workflow level progress is based on tasks counting. It is measured by completed tasks count (which will be assigned to *value*) against total tasks count (which will be assigned to *maximum*) for an active workflow. Thus workflow level progress measure is always based on steps.

Task progress is actually part of a workflow progress. It is that ask and workflow has two independent progress measurement methods.

There are two reasons to design a separate progress for task:

* + Long run tasks actually consume most time for workflows. In this case internal progress of a long run tasks is important than other workflow progress information.
	+ Workflow is organized in tasks, it will be more clear if workflow level progress includes only tasks completion information.

**2.4 CentOS Install Graph Progress Design**

**2.4.1 Install Task Milestone Events**

Besides task started and finished events, milestone events are created for task progress measurement. Take CentOS installation for example, 4 milestones is created for this tasks:

* + Profile downloaded: node rebooted and start to download kernel, RackHD monitor this events in install OS jobs.
	+ Kernel downloaded: installer kernel downloaded, a http GET operation is embedded in OS ipxe file to notification RackHD.
	+ Installation started: installer is started, a notification POST is embeded in kickstart file to notify RackHD.
	+ Installation finished: installation and configuration completed, a notification POST is embeded in kickstart file to notify RackHD.

A typical task progress information payload for profile downloaded will be:

progress: {
 value: 1,
 maximum: 4,
 description: 'Reboot suceeded, starting kernel download.',
 percentage: '25%'
}

Below pictures shows how milestone works and how progress message is generated during CentOS installation.



**2.4.2 CentOS Install Graph Progress Measurement**

CentOS install graph actually includes 4 tasks in sequnce:

* + set pxe boot, reboot, install os and callback.

Thus at workflow level CentOS install graph have 4 big steps for each tasks.

Besides install os, other tasks are non-long-run tasks and will only send two progress messages. There will be totally 14 progress messages during executing CentOS installation graph:

* + 4 task started and 4 task finished events
	+ 4 milestone progress messages for install tasks
	+ Graph started and finished events

Below pictures includes all 14 messages and display how two levels of progress works:



Other OS installations will have similar progress messages and measurements.

**2.5 Secure Erase Graph Progress Design**

**2.5.1 Secure Erase Timer Timeout Events.**

 Different from OS installation, secure erase job can get an accurate percentage number of erasing progress generated by erasing tools. The secure erase script will poll erasing progress based on a repeating 60 second timer, parse the percatege to get value and maximum, and send the payload to RackDH via notification API. Below is a typical secure erase progress information payload:

progress: {
 value: 3,
 maximum: 100,
 description: 'This is the 3th polling with 60s interval.',
 percentage: '3%'
}

Below pictures show how progress messages are generated during secure erase graph executing.



**2.4.2 CentOS Install Graph Progress Measurement**

Secure erase graph measurement is similar to CentOS installation except for:

* + Secure erase have 8 tasks. Secure erase task is the 4th task.
	+ Secure erase task progress is based on percentage not steps



**3. Issues and follow-ups**

* + **Extend task progress to other long run tasks**

Beside secure erase and OS installation, there are some other long run tasks:

* + Firmware update workflows
	+ Discovery workflow
	+ **Implement progress service**

Currently progress code is distributed to different RackHD services and even ipxe, kickstart file. We should consider putting all progress related code together to create an independent progress service that only responsible to report progress to users.