```
<#assign apiVersion = "2.0" />
<#assign messagesArchived = [] />
<#assign failedToUpdate = [] />
<#assign failedToUpdateErrorMsg = [] />
<#assign processingTime = 0 />
<#assign queryLimit = 100 />
<#assign allowedRoles = ["Administrator","Label - Moderator","Label - Admin"] />

<#if user.registered >
   <#assign user_has_role = false />
   <#list restadmin("/users/id/${user.id?c}/roles").roles.role as role>
      <#if role.name?? && allowedRoles?seq_contains("${role.name}")>
         <#assign user_has_role = true />
         <#break>
      </#if>
   </#list>
      <#if user_has_role >
            <#-- Start Date -->
            <#if http.request.parameters.name.startDate??>
                  <#assign startDate = http.request.parameters.name.startDate />
            <#else>
                  <#assign startDate = "2004-06-03T18:07:10Z" />
            </#if>
            <#-- End Date -->
            <#if http.request.parameters.name.endDate??>
                  <#assign endDate = http.request.parameters.name.endDate />
            <#else>
                  <#assign endDate = "2004-07-03T18:07:10Z" />
            </#if>
            <#-- Final End Date (when cleaning up a big timeframe on small batches) -->
            <#-- This is a Workaround. The API was returning not very accurate results (apparently at some point it was using original post_date and not conversation.last_post_date) so we used this date to catch results that were out of the small batch boundries (for example, January 2015) but it still meets the condition for the intended time frame (the whole year). -->
            <#if http.request.parameters.name.batchEndDate??>
                  <#assign batchEndDate = http.request.parameters.name.batchEndDate />
            </#if>
            <#-- From Board -->
            <#if http.request.parameters.name.board??>
                  <#assign board = http.request.parameters.name.board />
            <#else>
                  <#assign board = "" />
            </#if>
            <#-- Board ID for the correct archive board -->
            <#if http.request.parameters.name.moveTo??>
                  <#assign moveTo = http.request.parameters.name.moveTo />
            <#else>
                  <#assign moveTo = "" />
            </#if>
            <#-- Check if we actually want to move not solved threads only -->
            <#if http.request.parameters.name.notSolvedOnly??>
                  <#assign notSolvedOnly = http.request.parameters.name.notSolvedOnly?lower_case?boolean />
            <#else>
```

```
        <#assign notSolvedOnly = false />
    </#if>
    <#-- Check if we actually want to move the threads or just generate a report -->
    <#if http.request.parameters.name.moveThreads??>
        <#assign moveThreads = http.request.parameters.name.moveThreads />
    <#else>
        <#assign moveThreads = "false" />
    </#if>

    <#if board != "" && moveTo != "" >
        <#attempt>
            <#assign processingStartTime = .now />
            <#assign notSolvedQuery = "">
            <#if notSolvedOnly><#assign notSolvedQuery = " AND conversation.solved = false" /></#if>

            <#assign postCount = rest(apiVersion, "/search?q=" + "SELECT count(*) FROM messages
WHERE conversation.last_post_time <= ${endDate} AND conversation.last_post_time > ${startDate} AND depth = 0
AND board.id='${board}'${notSolvedQuery}"?url).data.count />
            <#assign resultPages = (postCount/queryLimit)?int+1 />
            <#if postCount gt 0>
                <#list 0..resultPages as page>
                    <#assign postsToArchive = rest(apiVersion, "/search?q=" + "SELECT id, board,
conversation, view_href, metrics, post_time FROM messages WHERE conversation.last_post_time <= ${endDate}
AND conversation.last_post_time > ${startDate} AND depth = 0 AND board.id='${board}'${notSolvedQuery}
ORDER BY conversation.last_post_time DESC LIMIT ${queryLimit} OFFSET ${page*queryLimit}"?url).data.items
/>
                    <#list postsToArchive as post>
                        <#attempt>
                            <#if (batchEndDate?? && post.conversation.last_post_time?long lte
batchEndDate?datetime.iso?long) || (post.conversation.last_post_time?long lte endDate?datetime.iso?long)>
                                <#attempt>
                                    <#if moveThreads == "true">
                                        <#assign apiResults =
rest('/messages/id/${post.id}/tagging/tags/add?tag.add=Archived') />
                                        <#assign apiResults =
rest('/messages/id/${post.id}/tagging/tags/add?tag.add=source-board-id:${post.board.id}') />
                                        <#assign apiResults =
rest('/messages/id/${post.id}/move/board/id/${moveTo}?
move_message.include_replies=true&move_message.ignore_notification=true') />
                                    </#if>
                                    <#assign messagesArchived = messagesArchived +
[{"messageID":"${post.id}","messageURL":"${asset.get(post.view_href)}","views":"${post.metrics.views}","movedFr
om":"${board}","movedTo":"${moveTo}","originalPostTime":"${post.post_time?
datetime}","lastPostTime":"${post.conversation.last_post_time?datetime}"}] />
                                <#recover>
                                    <#assign failedToUpdate = failedToUpdate + ["${post.id}"] />
                                    <#assign failedToUpdateErrorMsg = failedToUpdateErrorMsg +
[{"messageID":"${post.id}","errorMsg":"Error: An error might have ocurred when trying to move this
thread","threadDetails":"conversation.last_post_time = ${post.conversation.last_post_time?datetime?iso_local}; End
Date = ${endDate?datetime.iso}; Batch end date = ${batchEndDate!'Not provided'}; ${.error?json_string}"}] />
                                </#attempt>
                            <#else>
                                <#assign failedToUpdate = failedToUpdate + ["${post.id}"] />
                                <#assign failedToUpdateErrorMsg = failedToUpdateErrorMsg +
```

```
[{"messageID":"${post.id}","errorMsg":"Error: conversation.last_post_time is out of the time
range.","threadDetails":"conversation.last_post_time = ${post.conversation.last_post_time?datetime?iso_local}; End
Date = ${endDate?datetime.iso}; Batch end date = ${batchEndDate!'Not provided'}"}] />
                                        </#if>
                                    <#recover>
                                        <#assign failedToUpdateErrorMsg = failedToUpdateErrorMsg +
[{"messageID":"${post.id}","errorMsg":"Error: ${.error?json_string}","threadDetails":"conversation.last_post_time =
${(post.conversation.last_post_time?datetime)!'empty'}"}] />
                                    </#attempt>
                                </#list>
                            </#list>
                        </#if>
                        <#assign statusMessage = "Success" />
                        <#assign processingEndTime = .now />
                        <#-- <#assign processingTime = (processingEndTime?long - processingStartTime?long)/1000 /> -->
                    <#recover>
                        <#assign statusMessage = "Error: ${.error?json_string}" />
                    </#attempt>
                <#else>
                    <#assign statusMessage = "Error: Origin/Destination boards parameters can't be empty" />
                </#if>
            <#else>
                <#assign statusMessage = "Error: You need Administrator permissions to run this script" />
            </#if>
        <#else>
            <#assign statusMessage = "Error: You need Administrator permissions to run this script" />
        </#if>
        {
            "response": {
                "responseStatus" : "${statusMessage}",
                "errorCount" : "${failedToUpdateErrorMsg?size}",
                "endDate" : "${endDate!""}",
                "startDate" : "${startDate!""}",
                "sourceBoard" : "${board!""}",
                "archiveBoard" : "${moveTo!""}",
                "processingTime" : "${((.now?long - processingStartTime?long)/1000)!""} seconds",
                "processedPostCount" : "${postCount!""}",
                "messagesArchivedCount" : "${messagesArchived?size}",
                "messagesArchived" : [<#list messagesArchived as item>
{"messageID":"${item.messageID}","messageURL":"${item.messageURL}","views":"${item.views}","movedFrom":"
${item.movedFrom}","movedTo":"${item.movedTo}","originalPostTime":"${item.originalPostTime}","lastPostTime":
"${item.lastPostTime}"}<#if item?has_next>,</#if></#list>],
                "failedToUpdateCount" : "${failedToUpdate?size}",
                "failedToUpdate" : [<#list failedToUpdate as item>"${item}"<#if item?has_next>,</#if></#list>],
                "failedToUpdateErrorMsg" : [<#list failedToUpdateErrorMsg as item>{<#list item?keys as key>"${key}":
<#if item["${key}"]?is_string>"${item["${key}"]}"<#else>${item["${key}"]}</#if><#if key?has_next>,</#if>
</#list>}<#if item?has_next>,</#if></#list>],
                "requestURL" : "${http.request.url}"
            }
        }
```