Package ‘pyMTurkR’

February 4, 2020

Type Package
Title A Client for the 'MTurk' Requester API
Version 1.1.4
Maintainer Tyler Burleigh <tylerburleigh@gmail.com>
Description Provides access to the latest 'Amazon Mechanical Turk' ('MTurk') <https://www.mturk.com> Requester API (version '2017–01–17'), replacing the now deprecated 'MTurkR' package.
License GPL-2
Encoding UTF-8
LazyData true
Imports reticulate, curl, stats, utils, XML
RoxygenNote 6.1.1
Suggests testthat (>= 2.1.0), covr, knitr, rmarkdown
NeedsCompilation no
Author Tyler Burleigh [aut, cre] (https://twitter.com/tylerburleigh),
Thomas J. Leeper [aut] (<https://orcid.org/0000-0003-4097-6326>),
Solomon Messing [ctb],
Sean Murphy [ctb],
Jonathan Chang [ctb],
Andrew Brown [ctb]
Repository CRAN
Date/Publication 2020-02-04 13:10:02 UTC

R topics documented:

pyMTurkR-package .................................................. 3
AccountBalance .................................................... 4
ApproveAssignment ............................................... 5
AssignQualification ............................................. 6
BlockWorker ......................................................... 8
ChangeHITType ..................................................... 9
R topics documented:

- ContactWorker .................................................... 12
- CreateHIT .......................................................... 14
- CreateQualificationType ................................. 16
- DisableHIT .......................................................... 18
- DisposeQualificationType ......................... 20
- emptydf ............................................................. 21
- ExtendHIT .......................................................... 22
- GenerateExternalQuestion ......................... 24
- GenerateHITReviewPolicy ......................... 25
- GenerateHITsFromTemplate ........................... 28
- GenerateHTMLQuestion ....................................... 30
- GenerateNotification ............................................. 32
- GenerateQualificationRequirement ............... 33
- GetAssignment .................................................. 34
- GetBonuses ....................................................... 36
- GetClient .......................................................... 37
- GetHIT ............................................................. 38
- GetHITsForQualificationType ......................... 39
- GetQualificationRequests .............................. 41
- GetQualifications ................................................. 42
- GetQualificationScore ......................................... 44
- GetQualificationType ............................................. 45
- GetReviewableHITs ............................................... 46
- GetReviewResultsForHIT ......................................... 47
- GrantBonus ......................................................... 48
- GrantQualification .............................................. 50
- RegisterHITType ................................................... 51
- RejectAssignment ................................................. 53
- RevokeQualification ............................................. 54
- SearchHITs .......................................................... 56
- SearchQualificationTypes ............................... 57
- seconds ............................................................. 58
- SendTestEventNotification ......................... 59
- SetHITAsReviewing ............................................... 60
- SetHITTypeNotification ......................................... 62
- ToDataFrameAssignment ........................................ 63
- ToDataFrameBonusPayments ............................ 64
- ToDataFrameHITs .................................................... 64
- ToDataFrameQualificationRequests ............... 64
- ToDataFrameQualificationRequirements ........... 65
- ToDataFrameQualifications ............................. 65
- ToDataFrameQualificationTypes ....................... 65
- ToDataFrameQuestionFormAnswers .................. 66
- ToDataFrameReviewableHITs ......................... 66
- ToDataFrameReviewResults .............................. 67
- ToDataFrameWorkerBlock ............................... 67
- UpdateQualificationScore ......................... 67
- UpdateQualificationType ................................. 69
Description

This package provides access to the Amazon Mechanical Turk (MTurk) Requester API. The package provides users of the MTurk Requester User Interface with access to a variety of functions currently unavailable to them (the creation and maintenance of worker Qualifications, email notifications to workers through `ContactWorker`, automated reviewing of assignments using Review Policies, and streamlined bonus payments through `GrantBonus`). It also provides users with all functions available in the RUI directly in R as well as a large number of other functions, and a simple, interactive command-line tool for performing many operations.

Most users will find themselves using three principal functions: `CreateHIT`, `GetAssignments`, and `ApproveAssignments`, to create one or more HITs on the MTurk server, to retrieve completed assignments, and to approve assignments (and thus pay workers), respectively. As task complexity increases, additional functions are provided to handle worker qualifications, bonuses, emails to workers, automated review policies, bulk creation of HITs, and so forth.

Critically important, nothing in pyMTurkR will work during a given session without either first setting AWS credentials. The easiest way to do this is to specify `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables using `Sys.setenv()` or by placing these values in an `.Renviron` file. Credentials can also be specified in an AWS CLI credentials file as described here.

This package is a reboot of the MTurkR package after the MTurk API was updated in June 2019 and rendered it obsolete. This package uses reticulate to wrap boto3, the AWS SDK for Python, and access the MTurk API functions.

Author(s)

Tyler Burleigh

Maintainer: Tyler Burleigh <tylerburleigh@gmail.com>

References

Amazon Mechanical Turk

Amazon Mechanical Turk API Documentation

See Also

To get started using pyMTurkR, see the documentation for `CreateHIT` (for creating single tasks). For some tutorials on how to use MTurkR for specific use cases, see the following:
AccountBalance

Retrieves the amount of money (in US Dollars) in your MTurk account.

Usage

AccountBalance()

Details

AccountBalance takes no arguments.

accountbalance(), get_account_balance() and getbalance() are aliases for AccountBalance.

Value

Returns a list of length 2: “AvailableBalance”, the balance of the account in US Dollars, and “RequestMetadata”, the metadata for the request. Note: list is returned invisibly.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

MTurk Pricing Structure

Examples

```r
## Not run:
AccountBalance()

## End(Not run)
```
ApproveAssignment

ApproveAssignment
Approve Assignment(s)

Description
Approve one or more submitted assignments, or approve all assignments for a given HIT or HIT-Type. Also allows you to approve a previously rejected assignment. This function spends money from your MTurk account.

Usage
ApproveAssignment(assignments, feedback = NULL, rejected = FALSE, verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments
assignments A character string containing an AssignmentId, or a vector of multiple character strings containing multiple AssignmentIds, to approve.
feedback An optional character string containing any feedback for a worker. This must have length 1 or length equal to the number of workers. Maximum of 1024 characters.
rejected A logical indicating whether the assignment(s) had previously been rejected (default FALSE), or a vector of logicals.
verbose Optionally print the results of the API request to the standard output. Default is taken from getOption(‘pyMTurkR.verbose’,TRUE).

Details
Approve assignments, by AssignmentId (as returned by GetAssignment or by HITId or HITTypeId. Must specify assignments. ApproveAllAssignments approves all assignments of a given HIT or HITType without first having to perform GetAssignment.
ApproveAssignments() and approve() are aliases for ApproveAssignment. approveall() is an alias for ApproveAllAssignments.

Value
A data frame containing the list of AssignmentIds, feedback (if any), whether previous rejections were to be overridden, and whether or not each approval request was valid.

Author(s)
Tyler Burleigh, Thomas J. Leeper

References
API Reference: Approve Assignment
API Reference: Approve Rejected Assignment
AssignQualification

See Also

RejectAssignment

Examples

## Not run:
# Approve one assignment
ApproveAssignment(assignments = "26XXH0JPPSI23H54YVG7BKLEXAMPLE")

# Approve multiple assignments with the same feedback
ApproveAssignment(assignments = c("26XXH0JPPSI23H54YVG7BKLEXAMPLE1",
                                  "26XXH0JPPSI23H54YVG7BKLEXAMPLE2"),
                   feedback = "Great work!")

## End(Not run)

AssignQualification Assign Qualification

Description

Assign a Qualification to one or more workers. The QualificationType should have already been created by CreateQualificationType, or the details of a new QualificationType can be specified atomically. This function also provides various options for automatically specifying the value of a worker’s QualificationScore based upon a worker’s statistics.

Usage

AssignQualification(qual = NULL, workers, value = 1, notify = FALSE,
                   name = NULL, description = NULL, keywords = NULL, status = NULL,
                   retry.delay = NULL, test = NULL, answerkey = NULL,
                   test.duration = NULL, auto = NULL, auto.value = NULL,
                   verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qual</td>
<td>A character string containing a QualificationTypeId.</td>
</tr>
<tr>
<td>workers</td>
<td>A character string containing a WorkerId, or a vector of character strings containing multiple WorkerIds.</td>
</tr>
<tr>
<td>value</td>
<td>A character string containing the value to be assigned to the worker(s) for the QualificationType.</td>
</tr>
<tr>
<td>notify</td>
<td>A logical indicating whether workers should be notified that they have been assigned the qualification. Default is FALSE.</td>
</tr>
<tr>
<td>name</td>
<td>An optional character string specifying a name for a new QualificationType. This is visible to workers. Cannot be modified by UpdateQualificationType.</td>
</tr>
</tbody>
</table>
description: An optional character string specifying a longer description of the QualificationType. This is visible to workers. Maximum of 2000 characters.

keywords: An optional character string containing a comma-separated set of keywords by which workers can search for the QualificationType. Cannot be modified by UpdateQualificationType. Maximum of 1000 characters.

status: A character vector of “Active” or “Inactive”, indicating whether the QualificationType should be active and visible.

retry.delay: An optional time (in seconds) indicating how long workers have to wait before requesting the QualificationType after an initial rejection.

test: An optional character string consisting of a QuestionForm data structure, used as a test a worker must complete before the QualificationType is granted to them.

answerkey: An optional character string consisting of an AnswerKey data structure, used to automatically score the test.

test.duration: An optional time (in seconds) indicating how long workers have to complete the test.

auto: A logical indicating whether the Qualification is automatically granted to workers who request it. Default is FALSE.

auto.value: An optional parameter specifying the value that is automatically assigned to workers when they request it (if the Qualification is automatically granted).

verbose: Optionally print the results of the API request to the standard output. Default is taken from getOption('pyMTurkR.verbose', TRUE).

Details

A very robust function to assign a Qualification to one or more workers. The simplest use of the function is to assign a Qualification of the specified value to one worker, but assignment to multiple workers is possible. Workers can be assigned a Qualification previously created by CreateQualificationType, with the characteristics of a new QualificationType specified atomically, or a QualificationTypeID for a qualification created in the MTurk RUI. AssignQualifications(), assignqual() and AssociateQualificationWithWorker() are aliases.

Value

A data frame containing the list of workers, the QualificationTypeID, the value each worker was assigned, whether they were notified of their QualificationType assignment, and whether the request was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference
Examples

```r
## Not run:
qual1 <-
CreateQualificationType(name="Worked for me before",
    description="This qualification is for people who have worked for me before",
    status = "Active",
    keywords="Worked for me before")

# assign qualification to single worker
AssignQualification(qual1$QualificationTypeId, "A1RO9UJNXM6U65", value = "50")

# delete the qualification
DeleteQualificationType(qual1)

# assign a new qualification (defined atomically)
AssignQualification(workers = "A1RO9UJNXM6U65",
    name = "Worked for me before",
    description = "This qualification is for people who have worked for me before",
    status = "Active",
    keywords = "Worked for me before")

## End(Not run)
```

---

**BlockWorker**  
**Block Worker(s)**

**Description**

Block a worker. This prevents a worker from completing any HITs for you while they are blocked, but does not affect their ability to complete work for other requesters or affect their worker statistics.

**Usage**

```r
BlockWorker(workers, reasons = NULL,
    verbose =getOption("pyMTurkR.verbose", TRUE))
```

**Arguments**

- **workers**: A character string containing a WorkerId, or a vector of character strings containing multiple WorkerIds.
- **reasons**: A character string containing a reason for blocking a worker. This must have length 1 or length equal to the number of workers.
- **verbose**: Optionally print the results of the API request to the standard output. Default is taken from `getOption("pyMTurkR.verbose",TRUE)`. 

Details

BlockWorker prevents the specified worker from completing any of your HITs.

BlockWorkers(), block() and CreateWorkerBlock(), are aliases for BlockWorker. UnblockWorkers(), unblock(), and DeleteWorkerBlock() are aliases for UnblockWorker. blockedworkers() is an alias for GetBlockedWorkers.

Value

BlockWorker returns a data frame containing the list of workers, reasons (for blocking them), and whether the request to block was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference: Block

Examples

```r
## Not run:
BlockWorker("A1RO9UJNWXMU65", reasons="Did not follow HIT instructions.")
UnblockWorker("A1RO9UJNWXMU65")

## End(Not run)
```

---

ChangeHITType

Change HITType Properties of a HIT

Description

Change the HITType of a HIT from one HITType to another (e.g., to change the title, description, or qualification requirements associated with a HIT). This will cause a HIT to no longer be grouped with HITs of the previous HITType and instead be grouped with those of the new HITType. You cannot change the payment associated with a HIT without expiring the current HIT and creating a new one.

Usage

```r
ChangeHITType(hit = NULL, old.hit.type = NULL, new.hit.type = NULL,
title = NULL, description = NULL, reward = NULL, duration = NULL,
keywords = NULL, auto.approval.delay = as.integer(2592000),
qual.req = NULL, old.annotation = NULL,
verbose = getOption("pyMTurkR.verbose", TRUE))
```
Arguments

hit: An optional character string containing the HITId whose HITTypeId is to be changed, or a vector of character strings containing each of multiple HITIds to be changed. Must specify hit xor old.hit.type xor annotation.

old.hit.type: An optional character string containing the HITTypeId (or a vector of HITTypeId-Ids) whose HITs are to be changed to the new HITTypeId. Must specify hit xor old.hit.type xor annotation.

new.hit.type: An optional character string specifying the new HITTypeId that this HIT should be visibly grouped with (and whose properties, e.g. reward amount, this HIT should inherit).

title: An optional character string containing the title for the HITType. All HITs of this HITType will be visibly grouped to workers according to this title.

description: An optional character string containing a description of the HITType. This is visible to workers.

reward: An optional character string containing the per-assignment reward amount, in U.S. Dollars (e.g., “0.15”).

duration: An optional character string containing the duration of each HIT, in seconds (for example, as returned by seconds).

keywords: An optional character string containing a comma-separated set of keywords by which workers can search for HITs of this HITType.

auto.approval.delay: An optional character string specifying the amount of time, in seconds (for example, as returned by seconds), before a submitted assignment is automatically granted.

qual.req: An optional character string containing one a QualificationRequirement data structure, as returned by GenerateQualificationRequirement.

old.annotation: An optional character string specifying the value of the RequesterAnnotation field for a batch of HITs to change the HITType of. This can be used to change the HITType for all HITs from a “batch” created in the online Requester User Interface (RUI). To use a batch ID, the batch must be written in a character string of the form “BatchId:73832;”, where “73832” is the batch ID shown in the RUI. Must specify hit xor old.hit.type xor annotation.

verbose: Optionally print the results of the API request to the standard output. Default is taken from getOption(‘pyMTurkR.verbose’,TRUE).

Details

This function changes the HITType of a specified HIT (or multiple specific HITs or all HITs of a specified HITType) to a new HITType. hit xor old.hit.type must be specified. Then, either a new HITTypeId can be specified or a new HITType can be created by atomically by specifying the characteristics of the new HITType.

changehittype() and UpdateHITTypeOfHIT() are aliases.
Value

A data frame listing the HITId of each HIT who HITType was changed, its old HITTypeId and new HITTypeId, and whether the request for each HIT was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

See Also

CreateHIT
RegisterHITType

Examples

```r
## Not run:
hittype1 <- RegisterHITType(title = "10 Question Survey",
description = "Complete a 10-question survey about news coverage and your opinions",
reward = ".20",
duration = seconds(hours=1),
keywords = "survey, questionnaire, politics")
a <- GenerateExternalQuestion("https://www.example.com/", "400")
hit <- CreateHIT(hit.type = hittype1$HITTypeId,
assignments = 1,
expiration = seconds(days=1),
question = a$string)

# change to HITType with new reward amount
hittype2 <- RegisterHITType(title = "10 Question Survey",
description = "Complete a 10-question survey about news coverage and your opinions",
reward = ".45",
duration = seconds(hours=1),
keywords = "survey, questionnaire, politics")
ChangeHITType(hit = hit$HITId,
new.hit.type=hittype2$HITTypeId)

# Change to new HITType, with arguments stated atomically
ChangeHITType(hit = hit$HITId,
title = "10 Question Survey",
description = "Complete a 10-question survey about news coverage and your opinions",
reward = ".20",
duration = seconds(hours=1),
keywords = "survey, questionnaire, politics")

# expire and dispose HIT
ExpireHIT(hit = hit$HITId)
```
ContactWorker

DeleteHIT(hit = hit$HITId)

## End(Not run)

ContactWorker

Contact Worker(s)

Description

Contact one or more workers. This sends an email with specified subject line and body text to one or more workers. This can be used to recontact workers in panel/longitudinal research or to send follow-up work.

Usage

ContactWorker(subjects, msgs, workers, batch = FALSE, verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

subjects

A character string containing subject line of an email, or a vector of character strings of of length equal to the number of workers to be contacted containing the subject line of the email for each worker. Maximum of 200 characters.

msgs

A character string containing body text of an email, or a vector of character strings of of length equal to the number of workers to be contacted containing the body text of the email for each worker. Maximum of 4096 characters.

workers

A character string containing a WorkerId, or a vector of character strings containing multiple WorkerIds.

batch

A logical (default is FALSE), indicating whether workers should be contacted in batches of 100 (the maximum allowed by the API). This significantly reduces the time required to contact workers, but eliminates the ability to send customized messages to each worker.

verbose

Optionally print the results of the API request to the standard output. Default is taken from getOption("pyMTurkR.verbose",TRUE).

Details

Send an email to one or more workers, either with a common subject and body text or subject and body customized for each worker.

In batch mode (when batch=TRUE), workers are contacted in batches of 100 with a single identical email. If one email fails (e.g., for one worker) the other emails should be sent successfully. That is to say, the request as a whole will be valid but will return additional information about which workers were not contacted. This information can be found in the MTurkR log file and viewing the XML responses directly.
ContactWorker

Note: It is only possible to contact workers who have performed work for you previously. When attempting to contact a worker who has not worked for you before, this function will indicate that the request was successful even though the email is not sent. The function will return a value of "HardFailure" for Valid when this occurs. The printed results may therefore appear contradictory because MTurk reports that requests to contact these workers are Valid, but they are not actually contacted. In batch, this means that a batch will be valid but individual ineligible workers will be reported as not contacted.

ContactWorkers().contact().NotifyWorkers().NotifyWorker(). and notify() are aliases.

Value

A data frame containing the list of workers, subjects, and messages, and whether the request to contact each of them was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

Examples

```r
## Not run:
a <- "Complete a follow-up survey for $.50"
b <- "Thanks for completing my HIT! 
I will pay a $.50 bonus if you complete a follow-up survey by Friday at 5:00pm. 
The survey can be completed at http://www.surveymonkey.com/s/pssurvey?c=A1RO9UEXAMPLE."

d <- ContactWorker(subjects = a, 
                  msgs = b, 
                  workers = c1)

e <- ContactWorker(subjects = a, 
                   msgs = b, 
                   workers = c2, 
                   batch = TRUE)

## End(Not run)
```
CreateHIT

Create HIT

Description

Create a single HIT. This is the most important function in the package. It creates a HIT based upon the specified parameters: (1) characteristics inherited from a HITType or specification of those parameters and (2) some kind of Question data structure.

Usage

CreateHIT(hit.type = NULL, question = NULL, expiration, assignments = NULL, assignment.review.policy = NULL, hit.review.policy = NULL, annotation = NULL, unique.request.token = NULL, title = NULL, description = NULL, reward = NULL, duration = NULL, keywords = NULL, auto.approval.delay = NULL, qual.req = NULL, hitlayoutid = NULL, hitlayoutparameters = NULL, verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

hit.type An optional character string specifying the HITTypeId that this HIT should be generated from. If used, the HIT will inherit title, description, keywords, reward, and other properties of the HIT.

question A mandatory (unless layoutid is specified) character string containing a QuestionForm, HTMLQuestion, or ExternalQuestion data structure. In lieu of a question parameter, a hitlayoutid and, optionally, hitlayoutparameters can be specified.

expiration The time (in seconds) that the HIT should be available to workers. Must be between 30 and 31536000 seconds.

assignments A character string specifying the number of assignments

assignment.review.policy An optional character string containing an Assignment-level ReviewPolicy data structure as returned by GenerateAssignmentReviewPolicy.

hit.review.policy An optional character string containing a HIT-level ReviewPolicy data structure as returned by GenerateHITReviewPolicy.

annotation An optional character string annotating the HIT. This is not visible to workers, but can be used as a label by which to identify the HIT from the API.

unique.request.token An optional character string, included only for advanced users. It can be used to prevent creating a duplicate HIT. A HIT will not be created if a HIT was previously granted (within a short time window) using the same unique.request.token.
**CreateHIT**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>A character string containing the title for the HITType. All HITs of this HITType will be visibly grouped to workers according to this title. Maximum of 128 characters.</td>
</tr>
<tr>
<td>description</td>
<td>A character string containing a description of the HITType. This is visible to workers. Maximum of 2000 characters.</td>
</tr>
<tr>
<td>reward</td>
<td>A character string containing the per-assignment reward amount, in U.S. Dollars (e.g., “0.15”).</td>
</tr>
<tr>
<td>duration</td>
<td>A character string containing the amount of time workers have to complete an assignment for HITs of this HITType, in seconds (for example, as returned by <code>seconds</code>). Minimum of 30 seconds and maximum of 365 days.</td>
</tr>
<tr>
<td>keywords</td>
<td>An optional character string containing a comma-separated set of keywords by which workers can search for HITs of this HITType. Maximum of 1000 characters.</td>
</tr>
<tr>
<td>auto.approval.delay</td>
<td>An optional character string specifying the amount of time, in seconds (for example, as returned by <code>seconds</code>), before a submitted assignment is automatically granted. Maximum of 30 days.</td>
</tr>
<tr>
<td>qual.req</td>
<td>An optional list containing one or more QualificationRequirements, for example as returned by <code>GenerateQualificationRequirement</code>.</td>
</tr>
<tr>
<td>hitlayoutid</td>
<td>An optional character string including a HITLayoutId retrieved from a HIT “project” template generated in the Requester User Interface at '<a href="https://requester.mturk.com/create/projects">https://requester.mturk.com/create/projects</a>'. If the HIT template includes variable placeholders, must also specify hitlayoutparameters.</td>
</tr>
<tr>
<td>hitlayoutparameters</td>
<td>Required if using a hitlayoutid with placeholder values. This must be a list of lists containing Name and String values.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optionally print the results of the API request to the standard output. Default is taken from <code>getOption('pyMTurkR.verbose',TRUE)</code>.</td>
</tr>
</tbody>
</table>

**Details**

This function creates a new HIT and makes it available to workers. Characteristics of the HIT can either be specified by including a valid HITTypeId for “hit.type” or creating a new HITType by atomically specifying the characteristics of a new HITType.

When creating a HIT, some kind of Question data structure must be specified. Either, a QuestionForm, HTMLQuestion, or ExternalQuestion data structure can be specified for the question parameter or, if a HIT template created in the Requester User Interface (RUI) is being used, the appropriate hitlayoutid can be specified. If the HIT template contains variable placeholders, then the hitlayoutparameters should also be specified.

When creating a ExternalQuestion HITs, the `GenerateHITsFromTemplate` function can emulate the HIT template functionality by converting a template .html file into a set of individual HIT .html files (that would also have to be uploaded to a web server) and executing CreateHIT for each of these external files with an appropriate ExternalQuestion data structure specified for the question parameter.

Note that HIT and Assignment Review Policies are not currently supported.

`createhit()`, `create()`, `CreateHITWithHITType()`, and `createhitwithhittype()` are aliases.
CreateQualificationType

Value

A data frame containing the HITId and other details of the newly created HIT.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

Examples

```r
## Not run:
CreateHIT(title = "Survey",
   description = "5 question survey",
   reward = "0.10",
   assignments = 1,
   expiration = seconds(days = 4),
   duration = seconds(hours = 1),
   keywords = "survey, questionnaire",
   question = GenerateExternalQuestion("https://www.example.com/", "400"))
## End(Not run)
```

Description

Create a QualificationType. This creates a QualificationType, but does not assign it to any workers. All characteristics of the QualificationType (except name and keywords) can be changed later with UpdateQualificationType.

Usage

```r
CreateQualificationType(name, description, status, keywords = NULL,
   retry.delay = NULL, test = NULL, answerkey = NULL,
   test.duration = NULL, auto = NULL, auto.value = NULL,
   verbose = getOption("pyMTurkR.verbose", TRUE))
```
**CreateQualificationType**

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A name for the QualificationType. This is visible to workers. It cannot be modified by <code>UpdateQualificationType</code>.</td>
</tr>
<tr>
<td>description</td>
<td>A longer description of the QualificationType. This is visible to workers. Maximum of 2000 characters.</td>
</tr>
<tr>
<td>status</td>
<td>A character vector of “Active” or “Inactive”, indicating whether the QualificationType should be active and visible.</td>
</tr>
<tr>
<td>keywords</td>
<td>An optional character string containing a comma-separated set of keywords by which workers can search for the QualificationType. Maximum 1000 characters. These cannot be modified by <code>UpdateQualificationType</code>.</td>
</tr>
<tr>
<td>retry.delay</td>
<td>An optional time (in seconds) indicating how long workers have to wait before requesting the QualificationType after an initial rejection. If not specified, retries are disabled and Workers can request a Qualification of this type only once, even if the Worker has not been granted the Qualification.</td>
</tr>
<tr>
<td>test</td>
<td>An optional character string consisting of a QuestionForm data structure, used as a test a worker must complete before the QualificationType is granted to them.</td>
</tr>
<tr>
<td>answerkey</td>
<td>An optional character string consisting of an AnswerKey data structure, used to automatically score the test.</td>
</tr>
<tr>
<td>test.duration</td>
<td>An optional time (in seconds) indicating how long workers have to complete the test.</td>
</tr>
<tr>
<td>auto</td>
<td>A logical indicating whether the Qualification is automatically granted to workers who request it. Default is NULL meaning FALSE.</td>
</tr>
<tr>
<td>auto.value</td>
<td>An optional parameter specifying the value that is automatically assigned to workers when they request it (if the Qualification is automatically granted).</td>
</tr>
<tr>
<td>verbose</td>
<td>Optionally print the results of the API request to the standard output. Default is taken from <code>getOption('pyMTurkR.verbose',TRUE)</code>.</td>
</tr>
</tbody>
</table>

**Details**

A function to create a QualificationType. Active QualificationTypes are visible to workers and to other requesters. All characteristics of the QualificationType, other than the name and keywords, can later be modified by `UpdateQualificationType`. Qualifications can then be used to assign Qualifications to workers with `AssignQualification` and invoked as QualificationRequirements in `RegisterHITType` and/or `CreateHIT` operations.

`createqual()` is an alias.

**Value**

A data frame containing the QualificationTypeId and other details of the newly created QualificationType.

**Author(s)**

Tyler Burleigh, Thomas J. Leeper
## Not run:

# Create a Qualification Type
qual1 <- CreateQualificationType(name="Worked for me before",
                                 description="This qualification is for people who have worked for me before",
                                 status = "Active",
                                 keywords = "Worked for me before")
DisposeQualificationType(qual1$QualificationTypeId)

# Create a Qualification Type with a Qualification Test
f <- system.file("templates/qualificationtest1.xml", package = "pyMTurkR")
QuestionForm <- paste0(readLines(f, warn = FALSE), collapse = "")
qual2 <- CreateQualificationType(name = "Qual0001",
                                 description = "This is a qualification",
                                 status = "Active",
                                 test = QuestionForm,
                                 test.duration = 30)
DisposeQualificationType(qual2$QualificationTypeId)

# Create a Qualification Type with a Qualification Test and Answer Key
f <- system.file("templates/qualificationtest1.xml", package = "pyMTurkR")
QuestionForm <- paste0(readLines(f, warn = FALSE), collapse = "")
f <- system.file("templates/answerkey1.xml", package = "pyMTurkR")
AnswerKey <- paste0(readLines(f, warn = FALSE), collapse = "")
qual3 <- CreateQualificationType(name = "Qual0001",
                                 description = "This is a qualification",
                                 status = "Active",
                                 test = QuestionForm,
                                 test.duration = 30,
                                 answerkey = AnswerKey)
DisposeQualificationType(qual3$QualificationTypeId)

## End(Not run)
**DisableHIT**

**Description**
This function will allow you to expire a HIT early, which means it will no longer be available for new workers to accept. Optionally, when disabling the HIT you can approve all pending assignments and you can also try to delete the HIT.

**Usage**
```
DisableHIT(hit = NULL, hit.type = NULL, annotation = NULL,
    approve.pending.assignments = FALSE, skip.delete.prompt = FALSE,
    verbose = getOption("pyMTurkR.verbose", TRUE))
```

**Arguments**
- **hit**: A character string containing a HITId or a vector of character strings containing multiple HITIds. Must specify hit xor hit.type xor annotation.
- **hit.type**: An optional character string containing a HITTypeId (or a vector of HITTypeIds). Must specify hit xor hit.type xor annotation.
- **annotation**: An optional character string specifying the value of the RequesterAnnotation field for a batch of HITs. This can be used to disable all HITs from a “batch” created in the online Requester User Interface (RUI). To use a batch ID, the batch must be written in a character string of the form “BatchId:73832;”, where “73832” is the batch ID shown in the RUI. Must specify hit xor hit.type xor annotation.
- **approve.pending.assignments**: A logical indicating whether the pending assignments should be approved when the HIT is disabled.
- **skip.delete.prompt**: A logical indicating whether to skip the prompt that asks you to confirm the delete operation. If TRUE, you will not be asked to confirm that you wish to Delete the HITs. The prompt is a safeguard flag to protect the user from mistakenly deleting HITs.
- **verbose**: Optionally print the results of the API request to the standard output. Default is taken from `getOption('pyMTurkR.verbose', TRUE)`.

**Details**
Be careful when deleting a HIT: this will also delete the assignment data! Calling this function with `DeleteHIT()`, `deletehit()`, `DisposeHIT()`, or `disposehit()` will result in deleting the HIT. The user will be prompted before continuing, unless `skip.delete.prompt` is TRUE.

If you disable a HIT while workers are still working on an assignment, they will still be able to complete their task.

`DisposeHIT()`, `ExpireHIT()`, `DeleteHIT()`, `disablehit()`, `disposehit()`, `expirehit()`, `deletehit()` are aliases.

**Value**
A data frame containing a list of HITs and whether the request to disable each of them was valid.
DisposeQualificationType

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference: Update Expiration for HIT API Reference: Delete HIT

Examples

## Not run:

```r
# Disable a single HIT
hittype1 <- RegisterHITType(title = "10 Question Survey",
                             description = "Complete a 10-question survey",
                             reward = ".20",
                             duration = seconds(hours=1),
                             keywords = "survey, questionnaire, politics")

a <- GenerateExternalQuestion("https://www.example.com/", "400")
hit1 <- CreateHIT(hit.type = hittype1$HITTypeId,
                   assignments = 1,
                   expiration = seconds(days=1),
                   question = a$string)

DisableHIT(hit = hit1$HITId)

# Disable all HITs of a given HITType
DisableHIT(hit.type = hit1$HITTypeId)

# Disable all HITs of a given batch from the RUI
DisableHIT(annotation="BatchId:78382;")

# Delete the HIT previously disabled
DeleteHIT(hit = hit1$HITId)

## End(Not run)
```

DisposeQualificationType

Dispose QualificationType

Description

Dispose of a QualificationType. This deletes the QualificationType, Qualification scores for all workers, and all records thereof.

Usage

```r
DisposeQualificationType(qual, verbose = getOption("pyMTurkR.verbose", TRUE))
```
emptydf

Arguments

qual A character string containing a QualificationTypeId.
verbose Optionally print the results of the API request to the standard output. Default is taken from getOption('pyMTurkR.verbose', TRUE).

Details

A function to dispose of a QualificationType that is no longer needed. It will dispose of the Qualification type and any HIT types that are associated with it. It does not not revoke Qualifications already assigned to Workers. Any pending requests for this Qualification are automatically rejected. DisposeQualificationType(), disposequal(), and deletequal() are aliases.

Value

A data frame containing the QualificationTypeId and whether the request to dispose was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

Examples

## Not run:
qual1 <-
CreateQualificationType(name = "Worked for me before",
   description = "This qualification is for people who have worked for me before",
   status = "Active",
   keywords = "Worked for me before")
DisposeQualificationType(qual1$QualificationTypeId)

## End(Not run)

emptydf

Helper function that creates an empty data.frame

Description

Helper function that creates an empty data.frame

Usage

emptydf(nrow, ncol, names)
Arguments

- nrow: Number of rows
- ncol: Number of columns
- names: Number of names of the columns

---

Description

Extend the time remaining on a HIT or the number of assignments available for the HIT.

Usage

```r
ExtendHIT(hit = NULL, hit.type = NULL, annotation = NULL,
           add.assignments = NULL, add.seconds = NULL,
           unique.request.token = NULL, verbose = getOption("pyMTurkR.verbose", TRUE))
```

Arguments

- **hit**: An optional character string containing a HITId or a vector of character strings containing multiple HITIds. Must specify hit xor hit.type xor annotation.
- **hit.type**: An optional character string containing a HITTypeId (or a vector of HITType-Ids). Must specify hit xor hit.type xor annotation.
- **annotation**: An optional character string specifying the value of the RequesterAnnotation field for a batch of HITs. This can be used to extend all HITs from a “batch” created in the online Requester User Interface (RUI). To use a batch ID, the batch must be written in a character string of the form “BatchId:78382;”, where “73832” is the batch ID shown in the RUI. Must specify hit xor hit.type xor annotation.
- **add.assignments**: An optional character string containing the number of assignments to add to the HIT. Must be between 1 and 1000000000.
- **add.seconds**: An optional character string containing the amount of time to extend the HIT, in seconds (for example, returned by `seconds`). Must be between 1 hour (3600 seconds) and 365 days.
- **unique.request.token**: An optional character string, included only for advanced users.
- **verbose**: Optionally print the results of the API request to the standard output. Default is taken from `getOption('pyMTurkR.verbose', TRUE)`. 

ExtendHIT

Details

A useful function for adding time and/or additional assignments to a HIT. If the HIT is already expired, this reactivates the HIT for the specified amount of time. If all assignments have already been submitted, this reactivates the HIT with the specified number of assignments and previously specified expiration. Must specify a HITId xor a HITTypeId. If multiple HITs or a HITTypeId are specified, each HIT is extended by the specified amount.
extend() is an alias.

Value

A data frame containing the HITId, assignment increment, time increment, and whether each extension request was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference: Update Expiration API Reference: Create Additional Assignments for HIT

Examples

```r
## Not run:
a <- GenerateExternalQuestion("https://www.example.com/", "400")
hit1 <- CreateHIT(title = "Example",
    description = "Simple Example HIT",
    reward = ".01",
    expiration = seconds(days = 4),
    duration = seconds(hours = 1),
    keywords = "example",
    question = a$string)

# add assignments
ExtendHIT(hit = hit1$HITId, add.assignments = "20")

# add time
ExtendHIT(hit = hit1$HITId, add.seconds = seconds(days=1))

# add assignments and time
ExtendHIT(hit = hit1$HITId, add.assignments = "20", add.seconds = seconds(days=1))

# cleanup
DisableHIT(hit = hit1$HITId)

## End(Not run)
## Not run:
# Extend all HITs of a given batch from the RUI
ExtendHIT(abbreviation="BatchId:78382;", add.assignments = "20")
```
## GenerateExternalQuestion

*Generate ExternalQuestion*

### Description

Generate an `ExternalQuestion` data structure for use in the 'Question' parameter of the `CreateHIT` operation.

### Usage

```r
GenerateExternalQuestion(url, frame.height = 400)
```

### Arguments

- **url**: A character string containing the URL (served over HTTPS) of a HIT file stored anywhere other than the MTurk server.
- **frame.height**: A character string containing the integer value (in pixels) of the frame height for the `ExternalQuestion` iframe.

### Details

An `ExternalQuestion` is a HIT stored anywhere other than the MTurk server that is displayed to workers within an HTML iframe of the specified height. The URL should point to a page — likely an HTML form — that can retrieve several URL GET parameters for "AssignmentId" and "WorkerId", which are attached by MTurk when opening the URL.

Note: `url` must be HTTPS.

### Value

A list containing
- `xml.parsed`: an XML data structure
- `string`: an XML formatted as a character string
- `url.encoded`: character string containing a URL query parameter-formatted HTMLQuestion data structure for use in the `question` parameter of `CreateHIT`.

### Author(s)

Tyler Burleigh, Thomas J. Leeper

### References

API Reference
Generate HIT Review Policy

See Also

CreateHIT

Examples

```r
## Not run:
a <- GenerateExternalQuestion(url="http://www.example.com/", frame.height="400")

hit1 <-
CreateHIT(title = "Survey",
         description = "5 question survey",
         reward = ".10",
         expiration = seconds(days = 4),
         duration = seconds(hours = 1),
         keywords = "survey, questionnaire",
         question = a$string)

ExpireHIT(hit1$HITId)
DisposeHIT(hit1$HITId)

## End(Not run)
```

GenerateHITReviewPolicy

Generate HIT and/or Assignment Review Policies

Description

Generate a HIT Review Policy and/or Assignment Review Policy data structure for use in CreateHIT.

Usage

GenerateHITReviewPolicy(...)

Arguments

... ReviewPolicy parameters passed as named arguments.

Details

Converts a list of ReviewPolicy parameters into a ReviewPolicy data structure.

A ReviewPolicy works by testing whether an assignment or a set of assignments satisfies a particular condition. If that condition is satisfied, then specified actions are taken. Review Policies come in two “flavors”: Assignment-level Review Policies take actions based on “known” answers to questions in the HIT and HIT-level Review Policies take actions based on agreement among multiple
assignments. It is possible to specify both Assignment-level and HIT-level ReviewPolicies for the same HIT.

Assignment-level ReviewPolicies involve checking whether that assignment includes particular ("correct") answers. For example, an assignment might be tested to see whether a correct answer is given to one question by each worker as a quality control measure. The ReviewPolicy works by checking whether a specified percentage of known answers are correct. So, if a ReviewPolicy specifies two known answers for a HIT and the worker gets one of those known answers correct, the ReviewPolicy scores the assignment at 50 (i.e., 50 percent). The ReviewPolicy can then be customized to take three kinds of actions depending on that score: ApproveIfKnownAnswerScoreIsAtLeast (approve the assignment automatically), RejectIfKnownAnswerScoreIsLessThan (reject the assignment automatically), and ExtendIfKnownAnswerScoreIsLessThan (add additional assignments and/or time to the HIT automatically). The various actions can be combined to, e.g., both reject an assignment and add further assignments if a score is below the threshold, or reject below a threshold and approve above, etc.

HIT-level ReviewPolicies involve checking whether multiple assignments submitted for the same HIT “agree” with one another. Agreement here is very strict: answers must be exactly the same across assignments for them to be a matched. As such, it is probably only appropriate to use closed-ended (e.g., multiple choice) questions for HIT-level ReviewPolicies otherwise ReviewPolicy actions might be taken on irrelevant differences (e.g., word capitalization, spacing, etc.). The ReviewPolicy works by checking whether answers to multiple assignments are the same (or at least whether a specified percentage of answers to a given question are the same). For example, if the goal is to categorize an image into one of three categories, the ReviewPolicy will check whether two of three workers agree on the categorization (known as the “HIT Agreement Score”, which is a percentage of all workers who agree). Depending on the value of the HIT Agreement Score, actions can be taken. As of October 2014, only one action can be taken: ExtendIfHITAgreementScoreIsLessThan (extending the HIT in assignments by the number of assignments specified in ExtendMaximumAssignments or time as specified in ExtendMinimumTimeInSeconds).

Another agreement score (the “Worker Agreement Score”), measured the percentage of a worker’s responses that agree with other workers’ answers. Depending on the Worker Agreement Score, two actions can be taken: ApproveIfWorkerAgreementScoreIsAtLeast (to approve the assignment automatically) or RejectIfWorkerAgreementScoreIsLessThan (to reject the assignment automatically, with an optional reject reason supplied with RejectReason). A logical value (DisregardAssignmentIfRejected) specifies whether to exclude rejected assignments from the calculation of the HIT Agreement Score.

Note: An optional DisregardAssignmentIfKnownAnswerScoreIsLessThan excludes assignments if those assignments score below a specified “known” answers threshold as determined by a separate Assignment-level ReviewPolicy.

Value

A dictionary object HITReviewPolicy or AssignmentReviewPolicy.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference: QuestionForm
Examples

```r
## Not run:

# Generate a HIT Review Policy with GenerateHITReviewPolicy
lista <- list(QuestionIds = c("Question1", "Question2"),
              QuestionAgreementThreshold = 75,
              ApproveIfWorkerAgreementScoreIsAtLeast = 75,
              RejectIfWorkerAgreementScoreIsLessThan = 25)
policya <- do.call(GenerateHITReviewPolicy, lista)

# Manually define a HIT Review Policy
policya <- dict(
  list(
    'PolicyName' = 'SimplePlurality/2011-09-01',
    'Parameters' = list(
      dict(
        'Key' = 'QuestionIds',
        'Values' = list(
          'Question1',
          'Question2'
        )
      ),
      dict(
        'Key' = 'QuestionAgreementThreshold',
        'Values' = list(
          '75'
        )
      ),
      dict(
        'Key' = 'ApproveIfWorkerAgreementScoreIsAtLeast',
        'Values' = list(
          '75'
        )
      ),
      dict(
        'Key' = 'RejectIfWorkerAgreementScoreIsLessThan',
        'Values' = list(
          '25'
        )
      )
    )
  )
)
```
# Generate an Assignment Review Policy with GenerateAssignmentReviewPolicy

```r
listb <- list(AnswerKey = list("QuestionId1" = "B", "QuestionId2" = "A"),
              ApproveIfKnownAnswerScoreIsAtLeast = 99)
policyb <- do.call(GenerateAssignmentReviewPolicy, listb)
```

# Manually define an Assignment Review Policy

```r
policyb <- dict(
    list(
        dict(
            PolicyName = 'ScoreMyKnownAnswers/2011-09-01',
            Parameters = list(
                dict(
                    Key = 'AnswerKey',
                    MapEntries = list(
                        dict(
                            Key = 'QuestionId1',
                            Values = list('B')),
                        dict(
                            Key = 'QuestionId2',
                            Values = list('A')))
                ),
                dict(
                    Key = 'ApproveIfKnownAnswerScoreIsAtLeast',
                    Values = list('99'))
            ))
    ))
```

```r
## End(Not run)
```

---

**Generate HITs from a Template**

**Generate HITs from a Template**

**Description**

Generate individual HIT .html files from a local .html HIT template file, in the same fashion as the MTurk Requester User Interface (RUI).
GenerateHITsFromTemplate

Usage

GenerateHITsFromTemplate(template, input, filenames = NULL,
                         write.files = FALSE)

Arguments

template A character string or filename for an .html HIT template
input A data.frame containing one row for each HIT to be created and columns named identically to the placeholders in the HIT template file. Operation will fail if variable names do not correspond.
filenames An optional list of filenames for the HITs to be created. Must be equal to the number of rows in input.
write.files A logical specifying whether HIT .html files should be created and stored in the working directory. Or, alternatively, whether HITs should be returned as character vectors in a list.

Details

GenerateHITsFromTemplate generates individual HIT question content from a HIT template (containing placeholders for input data of the form ${variablename}). The tool provides functionality analogous to the MTurk RUI HIT template and can be performed on .html files generated therein. The HITs are returned as a list of character strings. If write.files = TRUE, a side effect occurs in the form of one or more .html files being written to the working directory, with filenames specified by the filenames option or, if filenames=NULL of the form “NewHIT1.html”, “NewHIT2.html”, etc.

Value

A list containing a character string for each HIT generated from the template.

Author(s)

Thomas J. Leeper

References

API Reference: Operation
API Reference: ExternalQuestion Data Structure

Examples

## Not run:
# create/edit template HTML file
# should have placeholders of the form `${varName}` for variable values
temp <- system.file("templates/htmlquestion2.xml", package = "pyMTurkR")
readLines(temp)

# create/load data.frame of template variable values
GenerateHTMLQuestion

Generate HTMLQuestion

Description

Generate an HTMLQuestion data structure for use in the ‘Question’ parameter of CreateHIT.

Usage

GenerateHTMLQuestion(character = NULL, file = NULL, frame.height = 450)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>An optional character string from which to construct the HTMLQuestion data structure.</td>
</tr>
<tr>
<td>file</td>
<td>An optional character string containing a filename from which to construct the HTMLQuestion data structure.</td>
</tr>
<tr>
<td>frame.height</td>
<td>A character string containing the integer value (in pixels) of the frame height for the HTMLQuestion iframe.</td>
</tr>
</tbody>
</table>
Details

Must specify either character or file.

To be valid, an HTMLQuestion data structure must be a complete XHTML document, including doctype declaration, head and body tags, and a complete HTML form (including the form tag with a submit URL, the assignmentId for the assignment as a form field, at least one substantive form field (can be hidden), and a submit button that posts to the external submit URL; see GenerateExternalQuestion). If you fail to include a complete form, workers will be unable to submit the HIT. See the API Documentation for a complete example.

MTurkR comes pre-installed with several simple examples of HTMLQuestion HIT templates, which can be found by examining the `templates` directory of the installed package directory. These examples include simple HTMLQuestion forms, as well as templates for categorization, linking to off-site surveys, and sentiment analysis. Note that the examples, while validated complete, do not include CSS styling.

Value

A list containing xmlparsed, an XML data structure, string, xml formatted as a character string, and urlencoded, character string containing a URL query parameter-formatted HTMLQuestion data structure for use in the question parameter of CreateHIT.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

See Also

CreateHIT
GenerateExternalQuestion

Examples

```r
## Not run:
f <- system.file("templates/htmlquestion1.xml", package = "pyMTurkR")
a <- GenerateHTMLQuestion(file=f)

hit1 <- CreateHIT(title = "Survey",
                 description = "5 question survey",
                 reward = ".10",
                 expiration = seconds(days = 4),
                 duration = seconds(hours = 1),
                 keywords = "survey, questionnaire",
                 question = a$string)
```
ExpireHIT(hit1$HITId)
DisposeHIT(hit1$HITId)

## End(Not run)

---

### GenerateNotification

**Description**

Generate a HITType Notification data structure for use in `SetHITTypeNotification`.

**Usage**

```r
GenerateNotification(destination, transport = "Email", event.type, version = "2006-05-05")
```

**Arguments**

- `destination`: Currently, a character string containing a complete email address (if `transport="Email"`), the SQS URL (if `transport="SQS"`) or the SNS topic (if `transport="SNS"`).
- `transport`: Only “Email”, “SQS” and “SNS” are supported. AWS recommends the use of the SQS transport.
- `event.type`: A character string containing one of: AssignmentAccepted, AssignmentAbandoned, AssignmentReturned, AssignmentSubmitted, AssignmentRejected, AssignmentApproved, HITCreated, HITExtended, HITDisposed, HITReviewable, HITCreated, HITExtended, HITDisposed, HITReviewable, HITExpired (the default), or Ping.
- `version`: Version of the HITType Notification API to use. Intended only for advanced users.

**Details**

Generate a Notification data structure for use in the `notification` option of `SetHITTypeNotification`.

**Value**

A dictionary object containing the Notification data structure.

**Author(s)**

Tyler Burleigh, Thomas J. Leeper

**References**

- API Reference
- API Reference: Concept
See Also

SetHITTypeNotification
SendTestEventNotification

GenerateQualificationRequirement

Description
Generate a QualificationRequirement data structure for use with CreateHIT or RegisterHITType.

Usage
GenerateQualificationRequirement(quals)

Arguments
quals A list of lists of Qualification parameters. Each list contains: QualificationTypeId (string, REQUIRED), Comparator (string, REQUIRED), LocaleValues (vector of integers), LocaleValues (list containing Country = string, and optionally Subdivision = string), RequiredToPreview (logical), ActionsGuarded (string). See example below.

Details
A convenience function to translate the details of a QualificationRequirement into the necessary structure for use in the qual.req parameter of CreateHIT or RegisterHITType. The function accepts a list of lists of Qualification parameters.

Value
Returns a special reticulated 'tuple' object

Author(s)
Tyler Burleigh, Thomas J. Leeper

References
API Reference

See Also
CreateHIT
RegisterHITType
## Examples

```r
## Not run:
quals.list <- list(
  list(QualificationTypeId = "2F1KVCNMHVMHV8E9PBUB2A4J79LU20F",
       Comparator = "Exists",
       IntegerValues = 1,
       RequiredToPreview = TRUE
  ),
  list(QualificationTypeId = "00000000000000000071",
       Comparator = "EqualTo",
       LocaleValues = list(Country = "US"),
       RequiredToPreview = TRUE
  )
)
GenerateQualificationRequirement(quals.list) -> qual.req

## End(Not run)
```

#### Description

Get an assignment or multiple assignments for one or more HITs (or a HITType) as a data frame.

#### Usage

```r
GetAssignment(assignment = NULL, hit = NULL, hit.type = NULL,
              annotation = NULL, status = NULL, results = as.integer(100),
              pagetoken = NULL, get.answers = FALSE, persist.on.error = FALSE,
              verbose = getOption("pyMTurkR.verbose", TRUE))
```

#### Arguments

- **assignment**: An optional character string specifying the AssignmentId of an assignment to return. Must specify assignment xor hit xor hit.type xor annotation.
- **hit**: An optional character string specifying the HITId whose assignments are to be returned, or a vector of character strings specifying multiple HITIds all of whose assignments are to be returned. Must specify assignment xor hit xor hit.type xor annotation.
- **hit.type**: An optional character string specifying the HITTypeId (or a vector of HITTypeId) of one or more HITs whose assignments are to be returned. Must specify assignment xor hit xor hit.type xor annotation.
annotation
An optional character string specifying the value of the RequesterAnnotation field for a batch of HITs. This can be used to retrieve all assignments for all HITs from a “batch” created in the online Requester User Interface (RUI). To use a batch ID, the batch must be written in a character string of the form “BatchId:73832;”, where “73832” is the batch ID shown in the RUI. Must specify assignment xor hit xor hit.type xor annotation.

status
An optional vector of character strings (containing one of more of “Approved”, “Rejected”, “Submitted”), specifying whether only a subset of assignments should be returned. If NULL, all assignments are returned (the default). Only applies when hit or hit.type are specified; ignored otherwise.

results
An optional character string indicating how many results to fetch per page. Must be between 1 and 100. Most users can ignore this.

pagetoken
An optional character string indicating which page of search results to start at. Most users can ignore this.

get.answers
An optional logical indicating whether to also get the answers. If TRUE, the returned object is a list with Assignments and Answers.

persist.on.error
A boolean specifying whether to persist on an error. Errors can sometimes happen when the server times-out, in cases where large numbers of Assignments are being retrieved.

verbose
Optionally print the results of the API request to the standard output. Default is taken from getOption('pymturkR.verbose', TRUE).

Details
This function returns the requested assignments. The function must specify an AssignmentId xor a HITId xor a HITTypeId. If an AssignmentId is specified, only that assignment is returned. If a HIT or HITType is specified, default behavior is to return all assignments through a series of sequential (but invisible) API calls meaning that returning large numbers of assignments (or assignments for a large number of HITs in a single request) may be time consuming.

GetAssignments(), assignment(), assignments(), and ListAssignmentsForHIT() are aliases.

Author(s)
Tyler Burleigh, Thomas J. Leeper

References
API Reference: GetAssignment
API Reference: ListAssignmentsForHIT

Examples

## Not run:
# get an assignment
GetAssignment(assignments = "26XXH@JPPSI23H54YVG78KLEXAMPLE")
GetBonuses

GetBonus Payments

Description
Get details of bonuses paid to workers, by HIT, HITType, Assignment, or Annotation.

Usage
GetBonuses(assignment = NULL, hit = NULL, hit.type = NULL, 
annotation = NULL, results = as.integer(100), pagetoken = NULL, 
verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>assignment</td>
<td>An optional character string containing an AssignmentId whose bonuses should be returned. Must specify assignment xor hit xor hit.type xor annotation.</td>
</tr>
<tr>
<td>hit</td>
<td>An optional character string containing a HITId whose bonuses should be returned. Must specify assignment xor hit xor hit.type xor annotation.</td>
</tr>
<tr>
<td>hit.type</td>
<td>An optional character string containing a HITTypeId (or a vector of HITType-Ids) whose bonuses should be returned. Must specify assignment xor hit xor hit.type xor annotation.</td>
</tr>
<tr>
<td>annotation</td>
<td>An optional character string specifying the value of the RequesterAnnotation field for a batch of HITs. This can be used to retrieve bonuses for all HITs from a “batch” created in the online Requester User Interface (RUI). To use a batch ID, the batch must be written in a character string of the form “BatchId:78382;”, where “78382” is the batch ID shown in the RUI. Must specify assignment xor hit xor hit.type xor annotation.</td>
</tr>
<tr>
<td>results</td>
<td>An optional character string indicating how many results to fetch per page. Must be between 1 and 100. Most users can ignore this.</td>
</tr>
<tr>
<td>pagetoken</td>
<td>An optional character string indicating which page of search results to start at. Most users can ignore this.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optionally print the results of the API request to the standard output. Default is taken from getOption(&quot;pyMTurkR.verbose&quot;, TRUE).</td>
</tr>
</tbody>
</table>
Details

Retrieve bonuses previously paid to a specified HIT, HITType, Assignment, or Annotation. bonuses(), getbonuses(), ListBonusPayments() and listbonuspayments() are aliases.

Value

A data frame containing the details of each bonus, specifically: AssignmentId, WorkerId, Amount, Reason, and GrantTime.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

See Also

GrantBonus

Examples

```r
## Not run:
# Get bonuses for a given assignment
GetBonuses(assignment = "26XXH0JPSSI23H54YVG7BKLO82DHNU")

# Get all bonuses for a given HIT
GetBonuses(hit = "2MQB727M0IGF304GJ16S1F4VE3AYDQ")

# Get bonuses from all HITs of a given batch from the RUI
GetBonuses(annotation = "BatchId:78382;">)

## End(Not run)
```

---

**GetClient**  
*Creates an MTurk Client using the AWS SDK for Python (Boto3)*

Description

Create an API client. Only advanced users will likely need to use this function. CheckAWSKeys() is a helper function that checks if your AWS keys can be found.

Usage

```r
GetClient(sandbox = getOption("pyMTurkR.sandbox", TRUE),
          restart.client = FALSE)
```
GetHIT

Arguments

sandbox
A logical indicating whether the client should be in the sandbox environment or the live environment.

restart.client
A boolean that specifies whether to force the creation of a new client.

Details

StartClient() is an alias

Author(s)

Tyler Burleigh

References

AWS SDK for Python (Boto3) Boto3 Docs

Examples

```r
## Not run:
GetClient()

## End(Not run)
```

Description

Retrieve various details of a HIT as a data frame.

Usage

GetHIT(hit, verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

hit
A character string specifying the HITId of the HIT to be retrieved.

verbose
Optionally print the results of the API request to the standard output. Default is taken from getOption('pyMTurkR.verbose',TRUE).

Details

GetHIT retrieves characteristics of a HIT. HITStatus is a wrapper that retrieves the Number of Assignments Pending, Number of Assignments Available, Number of Assignments Completed for the HIT(s), which is helpful for checking on the progress of currently available HITs. gethit() and hit() are aliases for GetHIT. status() is an alias for HITStatus.
Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

Examples

```r
## Not run:
# register HITType
hittype <-
RegisterHITType(title="10 Question Survey",
description="Complete a 10-question survey about news coverage and your opinions",
reward=".20",
duration=seconds(hours=1),
keywords="survey, questionnaire, politics")

da <- GenerateExternalQuestion("http://www.example.com/", "400")
hit1 <-
CreateHIT(hit.type = hittype$HITTypeId, question = a$string)

GetHIT(hit1$HITId)
HITStatus(hit1$HITId)

# cleanup
DisableHIT(hit1$HITId)

## End(Not run)
```

Description

Retrieve HITs according to the QualificationTypes that are required to complete those HITs.
GetHITsForQualificationType

Usage

GetHITsForQualificationType(qual, results = as.integer(100),
   pagetoken = NULL, verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

qual     A character string containing a QualificationTypeId.
results  An optional character string indicating how many results to fetch per page. Must
          be between 1 and 100. Most users can ignore this.
pagetoken An optional character string indicating which page of search results to start at.
            Most users can ignore this.
verbose  Optionally print the results of the API request to the standard output. Default is
            taken from getOption('pyMTurkR.verbose', TRUE).

Details

A function to retrieve HITs that require the specified QualificationType.
gethitsbyqual() and ListHITsForQualificationType() are aliases.

Value

A data frame containing the HITId and other requested characteristics of the qualifying HITs.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

Examples

## Not run:
GetHITsForQualificationType()

## End(Not run)
GetQualificationRequests

Get Qualification Requests

Description

Retrieve workers’ requests for a QualificationType.

Usage

GetQualificationRequests(qual = NULL, results = as.integer(100),
  pagetoken = NULL, verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

qual          An optional character string containing a QualificationTypeId to which the search
              should be restricted. If none is supplied, requests made for all Qualification-
              Types are returned.
results       An optional character string indicating how many results to fetch per page. Must
              be between 1 and 100. Most users can ignore this.
pagetoken     An optional character string indicating which page of search results to start at.
              Most users can ignore this.
verbose       Optionally print the results of the API request to the standard output. Default is
              taken from getOption('pyMTurkR.verbose', TRUE).

Details

A function to retrieve pending Qualification Requests made by workers, either for a specified
QualificationType or all QualificationTypes. Specifically, all active, custom QualificationTypes
are visible to workers, and workers can request a QualificationType (e.g., when a HIT requires
one they do not have). This function retrieves those requests so that they can be granted (with
GrantQualification) or rejected (with RejectQualification).

qualrequests() and ListQualificationRequests() are aliases.

Value

A data frame containing the QualificationRequestId, WorkerId, and other information (e.g., Quali-
fication Test results) for each request.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference
See Also

- GrantQualification
- RejectQualification

Examples

```r
## Not run:
GetQualificationRequests()

# Search for qualifications you own, then get requests for one of the quals
SearchQualificationTypes(must.be.owner = TRUE, verbose = FALSE) -> quals
quals$QualificationTypeId[[1]] -> qual1
GetQualificationRequests(qual1)

## End(Not run)
```

---

GetQualifications  *Get Qualifications*

**Description**

Get all Qualifications of a particular QualificationType assigned to Workers.

**Usage**

```r
GetQualifications(qual, status = NULL, results = as.integer(100),
pageToken = NULL, verbose = getOption("pyMTurkR.verbose", TRUE))
```

**Arguments**

- `qual`  
  A character string containing a QualificationTypeId for a custom (i.e., not built-in) QualificationType.

- `status`  
  An optional character string specifying whether only “Granted” or “Revoked” Qualifications should be returned.

- `results`  
  An optional character string indicating how many results to fetch per page. Must be between 1 and 100. Most users can ignore this.

- `pagetoken`  
  An optional character string indicating which page of search results to start at. Most users can ignore this.

- `verbose`  
  Optionally print the results of the API request to the standard output. Default is taken from `getOption("pyMTurkR.verbose",TRUE)`. 
GetQualifications

Details

A function to retrieve Qualifications granted for the specified QualificationType. To retrieve a specific Qualification score (e.g., for one worker), use `GetQualificationScore`.

A practical use for this is with automatically granted QualificationTypes. After workers request and receive an automatically granted Qualification that is tied to one or more HITs, `GetQualifications` can be used to retrieve the WorkerIds for workers that are actively working on those HITs (even before they have submitted an assignment).

`getquals()` and `ListWorkersWithQualificationType()` are aliases.

Value

A data frame containing the QualificationTypeId, WorkerId, and Qualification scores of workers assigned the Qualification.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

See Also

`GetQualificationScore`

`UpdateQualificationScore`

Examples

```r
## Not run:
qual1 <- AssignQualification(workers = "A1ROBUNWWMU65",
     name = "Worked for me before",
     description = "This qualification is for people who have worked for me before",
     status = "Active",
     keywords = "Worked for me before")

GetQualifications(qual1$QualificationTypeId)
RevokeQualification(qual1$QualificationTypeId, qual1$WorkerId)
GetQualifications(qual1$QualificationTypeId, status="Revoked")

DisposeQualificationType(qual1$QualificationTypeId)

## End(Not run)
```
Get a Worker's Qualification Score

Description

Get a Worker’s score for a specific Qualification. You can only retrieve scores for custom QualificationTypes.

Usage

GetQualificationScore(qual, workers, verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

qual A character string containing a QualificationTypeId for a custom Qualification-Type.
workers A character string containing a WorkerId, or a vector of character strings containing multiple WorkerIds, whose Qualification Scores you want to retrieve.
verbose Optionally print the results of the API request to the standard output. Default is taken from getOption('"pyMTurkR.verbose",TRUE).

Details

A function to retrieve one or more scores for a specified QualificationType. To retrieve all Qualifications of a given QualificationType, use GetQualifications instead. Both qual and workers can be vectors. If qual is not length 1 or the same length as workers, an error will occur. qalscore() is an alias.

Value

A data frame containing the QualificationTypeId, WorkerId, time the qualification was granted, the Qualification score, a column indicating the status of the qualification, and a column indicating whether the API request was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

See Also

UpdateQualificationScore
GetQualifications
Examples

```r
## Not run:
qual1 <- AssignQualification(workers = "A1RO9UJNWXMU65",
name = "Worked for me before",
description = "This qualification is for people who have worked for me before",
status = "Active",
keywords = "Worked for me before")

GetQualificationScore(qual1$QualificationTypeId, qual1$WorkerId)

# cleanup
DisposeQualificationType(qual1$QualificationTypeId)

## End(Not run)
```

Description

Get the details of a Qualification Type.

Usage

```r
GetQualificationType(qual, verbose = getOption("pyMTurkR.verbose", TRUE))
```

Arguments

- `qual` A character string containing a QualificationTypeId.
- `verbose` Optionally print the results of the API request to the standard output. Default is taken from `getOption("pyMTurkR.verbose", TRUE)`.

Details

Retrieve characteristics of a specified QualificationType (as originally specified by `CreateQualificationType`). `qualtype()` is an alias.

Value

A data frame containing the QualificationTypeId of the newly created QualificationType and other details as specified in the request.

Author(s)

Tyler Burleigh, Thomas J. Leeper
References

API Reference

Examples

```r
## Not run:
qual1 <- CreateQualificationType(name="Worked for me before",
      description="This qualification is for people who have worked for me before",
      status = "Active",
      keywords="Worked for me before")
GetQualificationType(qual1$QualificationTypeId)
DisposeQualificationType(qual1$QualificationTypeId)

## End(Not run)
```

GetReviewableHITs  Get Reviewable HITs

Description

Get HITs that are currently reviewable.

Usage

```r
GetReviewableHITs(hit.type = NULL, status = "Reviewable",
                   results = as.integer(100), pagetoken = NULL,
                   verbose = getOption("pyMTurkR.verbose", TRUE))
```

Arguments

- **hit.type**: An optional character string containing a HITTypeId to consider when looking for reviewable HITs.
- **status**: An optional character string of either “Reviewable” or “Reviewing” limiting the search to HITs of with either status.
- **results**: An optional character string indicating how many results to fetch per page. Must be between 1 and 100. Most users can ignore this.
- **pagetoken**: An optional character string indicating which page of search results to start at. Most users can ignore this.
- **verbose**: Optionally print the results of the API request to the standard output. Default is taken from `getOption('pyMTurkR.verbose', TRUE)`.

Details

A simple function to return the HITIds of HITs currently in “Reviewable” or “Reviewing” status. To retrieve additional details about each of these HITs, see `GetHIT`. This is an alternative to `SearchHITs`. `reviewable()` is an alias.
GetReviewResultsForHIT

Value

A data frame containing HITIds and Requester Annotations.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

Examples

```r
## Not run:
GetReviewableHITs()

## End(Not run)
```

GetReviewResultsForHIT

*Get ReviewPolicy Results for a HIT*

Description

Get HIT- and/or Assignment-level ReviewPolicy Results for a HIT

Usage

```r
GetReviewResultsForHIT(hit, policy.level = NULL, results = as.integer(100), pagetoken = NULL, verbose = getOption("pyMTurkR.verbose", TRUE))
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hit</td>
<td>A character string containing a HITId.</td>
</tr>
<tr>
<td>policy.level</td>
<td>Either HIT or Assignment. If NULL (the default), all data for both policy levels is retrieved.</td>
</tr>
<tr>
<td>results</td>
<td>An optional character string indicating how many results to fetch per page. Must be between 1 and 100. Most users can ignore this.</td>
</tr>
<tr>
<td>pagetoken</td>
<td>An optional character string indicating which page of search results to start at. Most users can ignore this.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optionally print the results of the API request to the standard output. Default is taken from getOption(&quot;pyMTurkR.verbose&quot;,TRUE).</td>
</tr>
</tbody>
</table>
Details

A simple function to return the results of a ReviewPolicy. This is intended only for advanced
users, who should reference MTurk documentation for further information or see the notes in
GenerateHITReviewPolicy.

reviewresults and ListReviewPolicyResultsForHIT are aliases.

Value

A four-element list containing up to four named data frames, depending on what ReviewPol-
icy (or ReviewPolicies) were attached to the HIT and whether results or actions are requested:
AssignmentReviewResult, AssignmentReviewAction, HITReviewResult, and/or HITReviewAction.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference
API Reference (ReviewPolicies)
API Reference (Data Structure)

See Also

CreateHIT
GenerateHITReviewPolicy

GrantBonus Pay Bonus to Worker

Description

Pay a bonus to one or more workers. This function spends money from your MTurk account and
will fail if insufficient funds are available.

Usage

GrantBonus(workers, assignments, amounts, reasons, skip.prompt = FALSE,
unique.request.token = NULL, verbose = getOption("pyMTurkR.verbose",
TRUE))
**Arguments**

- **workers**: A character string containing a WorkerId, or a vector of character strings containing multiple WorkerIds.
- **assignments**: A character string containing an AssignmentId for an assignment performed by that worker, or a vector of character strings containing the AssignmentId for an assignment performed by each of the workers specified in workers.
- **amounts**: A character string containing an amount (in U.S. Dollars) to bonus the worker(s), or a vector (of length equal to the number of workers) of character strings containing the amount to be paid to each worker.
- **reasons**: A character string containing a reason for bonusing the worker(s), or a vector (of length equal to the number of workers) of character strings containing the reason to bonus each worker. The reason is visible to each worker and is sent via email.
- **skip.prompt**: A logical indicating whether to skip the prompt that asks you to continue when duplicate AssignmentIds are found. If TRUE, you will not be asked to confirm. The prompt is a safeguard flag to protect the user from mistakenly paying a bonus twice.
- **unique.request.token**: An optional character string, included only for advanced users. It can be used to prevent resending a bonus. A bonus will not be granted if a bonus was previously granted (within a short time window) using the same unique.request.token.
- **verbose**: Optionally print the results of the API request to the standard output. Default is taken from `getOption('pymTurkR.verbose',TRUE)`.

**Details**

A simple function to grant a bonus to one or more workers. The function is somewhat picky in that it requires a WorkerId, the AssignmentId for an assignment that worker has completed, an amount, and a reason for the bonus, for each bonus to be paid. Optionally, the amount and reason can be specified as single (character string) values, which will be used for each bonus. `bonus()`, `paybonus()`, and `sendbonus()` are aliases.

**Value**

A data frame containing the WorkerId, AssignmentId, amount, reason, and whether each request to bonus was valid.

**Author(s)**

Tyler Burleigh, Thomas J. Leeper

**References**

API Reference

**See Also**

GetBonuses
Examples

## Not run:
# Grant a single bonus
a <- "A1RO9UEXAMPLE"
b <- "26XXH0JPPSI23H54YVG7BKLEEXAMPLE"
c <- ".50"
d <- "Thanks for your great work on my HITs!
Hope to work with you, again!"
GrantBonus(workers=a, assignments=b, amounts=c, reasons=d)

## End(Not run)

## Not run:
# Grant bonuses to multiple workers
a <- c("A1RO9EXAMPLE1","A1RO9EXAMPLE2","A1RO9EXAMPLE3")
b <-
c("26XXH0JPPSI23H54YVG7BKLEEXAMPLE1",
  "26XXH0JPPSI23H54YVG7BKLEEXAMPLE2",
  "26XXH0JPPSI23H54YVG7BKLEEXAMPLE3")
c <- c(".50",".10",".25")
d <- "Thanks for your great work on my HITs!"
GrantBonus(workers=a, assignments=b, amounts=c, reasons=d)

## End(Not run)

GrantQualification

Grant/Accept or Reject a Qualification Request

Description

Grant/accept or reject a worker’s request for a Qualification.

Usage

GrantQualification(qual.requests, values, reason = NULL,
verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

qual.requests  A character string containing a QualificationRequestId (for example, returned by GetQualificationRequests), or a vector of QualificationRequestId.
values  A character string containing the value of the Qualification to be assigned to the worker, or a vector of values of length equal to the number of QualificationRequests.
reason  An optional character string, or vector of character strings of length equal to length of the qual.requests parameter, supplying each worker with a reason for rejecting their request for the Qualification. Workers will see this message. Maximum of 1024 characters.
verbose

Optionally print the results of the API request to the standard output. Default is taken from \
ggetOption('pyMTurkR.verbose',TRUE).

Details

Qualifications are publicly visible to workers on the MTurk website and workers can request Qual-\nfications (e.g., when a HIT requires a QualificationType that they have not been assigned). Qualifi-\ncationRequests can be retrieved via GetQualificationRequests. GrantQualification grants the \nspecified qualification requests. Requests can be rejected with RejectQualifications.

Note that granting a qualification may have the consequence of modifying a worker’s existing quali-\nfication score. For example, if a worker already has a score of 100 on a given QualificationType and \nthen requests the same QualificationType, a GrantQualification action might increase or decrease that worker’s qualification score.

Similarly, rejecting a qualification is not the same as revoking a worker’s Qualification. For exam-\nple, if a worker already has a score of 100 on a given QualificationType and then requests the same QualificationType, a RejectQualification leaves the worker's existing Qualification in place. Use RevokeQualification to entirely remove a worker’s Qualification.

GrantQualifications(), grantqual(), AcceptQualificationRequest() and acceptrequest() are aliases; RejectQualifications() and rejectrequest() are aliases.

Value

A data frame containing the QualificationRequestId, reason for rejection (if applicable; only for RejectQualification), and whether each request was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference: AcceptQualificationRequest

See Also

GetQualificationRequests

Description

Register a HITType on MTurk, in order to create one or more HITs to show up as a group to workers.
RegisterHITType

Usage

RegisterHITType(title, description, reward, duration, keywords = NULL,
    auto.approval.delay = as.integer(2592000), qual.req = NULL,
    verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

title
A character string containing the title for the HITType. All HITs of this HITType
will be visibly grouped to workers according to this title. Maximum of 128
characters.

description
A character string containing a description of the HITType. This is visible to

reward
A character string containing the per-assignment reward amount, in U.S. Dollars
(e.g., “0.15”).

duration
A character string containing the amount of time workers have to complete an
assignment for HITs of this HITType, in seconds (for example, as returned by
seconds). Minimum of 30 seconds and maximum of 365 days.

keywords
An optional character string containing a comma-separated set of keywords by
which workers can search for HITs of this HITType. Maximum of 1000 charac-
ters.

auto.approval.delay
An optional character string specifying the amount of time, in seconds (for ex-
ample, as returned by seconds), before a submitted assignment is automatically
granted. Maximum of 30 days.

qual.req
An optional character string containing one or more QualificationRequirements
data structures, for example as returned by GenerateQualificationRequirement.

verbose
Optionally print the results of the API request to the standard output. Default is
taken from getOption('pyMTurkR.verbose', TRUE).

Details

All HITs of a given HITType are visibly grouped together for workers and share common properties
(e.g., reward amount, QualificationRequirements). This function registers a HITType in the MTurk
system, which can then be used when creating individual HITs. If a requester wants to change these
properties for a specific HIT, the HIT should be changed to a new HITType (see ChangeHITType).

hittype(), CreateHITType(), and createhittype() are aliases.

Value

A two-column data frame containing the HITTypeId of the newly registered HITType and an indi-
cator for whether the registration request was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper
RejectAssignment

References

API Reference: Operation

See Also

CreateHIT
ChangeHITType

Examples

## Not run:
RegisterHITType(title="10 Question Survey",
description="Complete a 10-question survey about news coverage and your opinions",
reward=".20",
duration=seconds(hours=1),
keywords="survey, questionnaire, politics")

## End(Not run)

---

RejectAssignment  Reject Assignment

Description

Reject a Worker’s assignment (or multiple assignments) submitted for a HIT. Feedback should be provided for why an assignment was rejected.

Usage

RejectAssignment(assignments, feedback, 
verbose =getOption("pyMTurkR.verbose", TRUE))

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>assignments</td>
<td>A character string containing an AssignmentId, or a vector of multiple character strings containing multiple AssignmentIds, to reject.</td>
</tr>
<tr>
<td>feedback</td>
<td>A character string containing any feedback for a worker. This must have length 1 or length equal to the number of workers.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optionally print the results of the API request to the standard output. Default is taken from getOption('pyMTurkR.verbose',TRUE).</td>
</tr>
</tbody>
</table>
RevokeQualification

Details

Reject assignments, by AssignmentId (as returned by GetAssignment). More advanced functionality to quickly reject many or all assignments (ala ApproveAllAssignments) is intentionally not provided.

RejectAssignments() and reject() are aliases.

Value

A data frame containing the list of AssignmentIds, feedback (if any), and whether or not each rejection request was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

See Also

ApproveAssignment

Examples

## Not run:
RejectAssignment(assignments = "26XXH0JPPSI23H54YVG7BKLEXAMPLE")

## End(Not run)

RevokeQualification       Revoke a Qualification from a Worker

Description

Revoke a Qualification from a worker or multiple workers. This deletes their qualification score and any record thereof.

Usage

RevokeQualification(qual, workers, reasons = NULL,
     verbose = getOption("pyMTurkR.verbose", TRUE))
**RevokeQualification**

**Arguments**

- `qual` A character string containing a QualificationTypeId.
- `workers` A character string containing a WorkerId, or a vector of character strings containing multiple WorkerIds.
- `reasons` An optional character string, or vector of character strings of length equal to length of the `workers` parameter, supplying each worker with a reason for revoking their Qualification. Workers will see this message.
- `verbose` Optionally print the results of the API request to the standard output. Default is taken from `getOption('pyMTurkR.verbose',TRUE)`.

**Details**

A simple function to revoke a Qualification assigned to one or more workers.

`RevokeQualifications()`, `revokequal()` and `DisassociateQualificationFromWorker()` are aliases.

**Value**

A data frame containing the QualificationTypeId, WorkerId, reason (if applicable), and whether each request was valid.

**Author(s)**

Tyler Burleigh, Thomas J. Leeper

**References**

API Reference

**See Also**

- `GrantQualification`
- `RejectQualification`

**Examples**

```r
## Not run:
qual1 <-
AssignQualification(workers = "A1RO9UJNWXMU65",
                     name = "Worked for me before",
                     description = "This qualification is for people who have worked for me before",
                     status = "Active",
                     keywords = "Worked for me before")

RevokeQualification(qual = qual1$QualificationTypeId,
                      worker = qual1$WorkerId,
                      reason = "No longer needed")
```
SearchHITs

SearchHITs(qual1$QualificationTypeId)

## End(Not run)

SearchHITs  Search your HITs

Description

Search for your HITs and return those HITs as R objects.

Usage

SearchHITs(return.pages = NULL, results = as.integer(100),
           pagetoken = NULL, verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

return.pages  An integer indicating how many pages of results should be returned.
results       An optional character string indicating how many results to fetch per page. Must
              be between 1 and 100. Most users can ignore this.
pagetoken     An optional character string indicating which page of search results to start at.
              Most users can ignore this.
verbose       Optionally print the results of the API request to the standard output. Default is
t               taken from getOption('pyMTurkR.verbose',TRUE).

Details

Retrieve your current HITs (and, optionally, characteristics thereof).

searchhits(), ListHITs(), and listhits() are aliases

Value

A list containing data frames of HITs and Qualification Requirements

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference
Examples

```r
## Not run:
SearchHITs()

## End(Not run)
```

Description

Search for Qualification Types.

Usage

```r
SearchQualificationTypes(search.query = NULL,  
must.be.requestable = FALSE, must.be.owner = FALSE,  
results = as.integer(100), return.pages = NULL, pagetoken = NULL,  
verbose = getOption("pyMTurkR.verbose", TRUE))
```

Arguments

- `search.query`: An optional character string to use as a search query
- `must.be.requestable`: A boolean indicating whether the Qualification must be requestable by Workers or not.
- `must.be.owner`: A boolean indicating whether to search only the Qualifications you own / created, or to search all Qualifications. Defaults to FALSE.
- `results`: An optional character string indicating how many results to fetch per page. Must be between 1 and 100. Most users can ignore this.
- `return.pages`: An integer indicating how many pages of results should be returned.
- `pagetoken`: An optional character string indicating which page of search results to start at. Most users can ignore this.
- `verbose`: Optionally print the results of the API request to the standard output. Default is taken from `getOption('pyMTurkR.verbose',TRUE)`.

Details

This function will search Qualification Types. It can search through the Qualifications you created, or through all the Qualifications that exist.

`SearchQuals()`, `searchquals()`, `ListQualificationTypes()` `listquals()`, `ListQuals()` are aliases
Value

A data frame of Qualification Types

Author(s)

Tyler Burleigh

References

API Reference

Examples

```r
## Not run:
SearchQuals()
## End(Not run)
```

### seconds

<table>
<thead>
<tr>
<th>seconds</th>
<th>Convert arbitrary times to seconds</th>
</tr>
</thead>
</table>

Description

A convenience function to convert arbitrary numbers of days, hours, minutes, and/or seconds into seconds.

Usage

seconds(days = NULL, hours = NULL, minutes = NULL, seconds = NULL)

Arguments

days An optional number of days.
hours An optional number of hours.
minutes An optional number of minutes.
seconds An optional number of seconds.

Details

A convenience function to convert arbitrary numbers of days, hours, minutes, and/or seconds into seconds. For example, to be used in setting a HIT expiration time. MTurk only accepts times (e.g., for HIT expirations, or the duration of assignments) in seconds. This function returns an integer value equal to the number of seconds of the input, and can be used atomically within other MTurkR calls (e.g., CreateHIT).
SendTestEventNotification

Value
An integer equal to the requested amount of time in seconds.

Author(s)
Thomas J. Leeper

Description
Test a HITType Notification, for example, to try out a HITType Notification before creating a HIT.

Usage
SendTestEventNotification(notification, test.event.type, verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments
- notification: A dictionary object Notification structure (e.g., returned by GenerateNotification).
- test.event.type: A character string containing one of: AssignmentAccepted, AssignmentAbandoned, AssignmentReturned, AssignmentSubmitted, AssignmentRejected, AssignmentApproved, HITCreated, HITExtended, HITDisposed, HITReviewable, HITCreated, HITExtended, HITDisposed, HITReviewable, HITExpired (the default), or Ping.
- verbose: Optionally print the results of the API request to the standard output. Default is taken from getOption('pyMTurkR.verbose',TRUE).

Details
Test a Notification configuration. The test mimics whatever the Notification configuration will do when the event described in test.event.type occurs. For example, if a Notification has been configured to send an email any time an Assignment is Submitted, testing for an AssignmentSubmitted event should trigger an email. Similarly, testing for an AssignmentReturned event should do nothing.

notification is an alias.

Value
A data frame containing the notification, the event type, and details on whether the request was valid. As a side effect, a notification will be sent to the configured destination (either an email or an SQS queue).
SetHITAsReviewing

Author(s)
Tyler Burleigh, Thomas J. Leeper

References
API Reference

See Also
SetHITTypeNotification

Examples

```r
## Not run:
hitype <- RegisterHITType(title="10 Question Survey",
                           description = "Complete a 10-question survey",
                           reward = ".20",
                           duration = seconds(hours = 1),
                           keywords = "survey, questionnaire, politics")

a <- GenerateNotification("requester@example.com", event.type = "HITExpired")

SetHITTypeNotification(hit.type = hitype$HITTypeId,
                        notification = a,
                        active = TRUE)

## End(Not run)
```

---

SetHITAsReviewing  Set HIT as “Reviewing”

Description
Update the review status of a HIT, from “Reviewable” to “Reviewing” or the reverse.

Usage

```r
SetHITAsReviewing(hit = NULL, hit.type = NULL, annotation = NULL,
                   revert = FALSE, verbose = getOption("pyMTurkR.verbose", TRUE))
```

Arguments

hit  An optional character string containing a HITId, or a vector character strings containing HITIds, whose status are to be changed. Must specify hit xor hit.type xor annotation.
hit.type  An optional character string specifying a HITTypeId (or a vector of HITTypeIds), all the HITs of which should be set as “Reviewing” (or the reverse). Must specify hit xor hit.type xor annotation.

annotation An optional character string specifying the value of the RequesterAnnotation field for a batch of HITs. This can be used to set the review status all HITs from a “batch” created in the online Requester User Interface (RUI). To use a batch ID, the batch must be written in a character string of the form “BatchId:78382;”, where “73832” is the batch ID shown in the RUI. Must specify hit xor hit.type xor annotation.

revert An optional logical to revert the HIT from “Reviewing” to “Reviewable”.

verbose Optionally print the results of the API request to the standard output. Default is taken from getOption("pyMTurkR.verbose",TRUE).

Details
A function to change the status of one or more HITs (or all HITs of a given HITType) to “Reviewing” or the reverse. This affects what HITs are returned by GetReviewableHITs. Must specify a HITId xor a HITTypeId xor an annotation.

reverting() and UpdateHITReviewStatus() are aliases.

Value
A data frame containing HITId, status, and whether the request to change the status of each was valid.

Author(s)
Tyler Burleigh, Thomas J. Leeper

References
API Reference

See Also
GetReviewableHITs

Examples

## Not run:

a <- GenerateExternalQuestion("https://www.example.com/", "400")
hit1 <- CreateHIT(hit.type = "2FFNCWYB49F9BBJWA4SJUNST5OF5OW",
   question = a$string,
   expiration = seconds(hours = 1))
SetHITAsReviewing(hit1$HITId)

# cleanup
SetHITTypeNotification

Configure a HITType Notification

Description

Configure a notification to be sent when specific actions occur for the specified HITType.

Usage

SetHITTypeNotification(hit.type, notification = NULL, active = NULL, verbose = getOption("pyMTurkR.verbose", TRUE))

Arguments

hit.type A character string specifying the HITTypeId of the HITType for which notifications are being configured.
notification An optional dictionary object Notification structure (e.g., returned by GenerateNotification).
active A logical indicating whether the Notification is active or inactive.
verbose Optionally print the results of the API request to the standard output. Default is taken from getOption('pyMTurkR.verbose', TRUE).

Details

Configure a notification to be sent to the requester whenever an event (specified in the Notification object) occurs. This is useful, for example, to enable email notifications about when assignments are submitted or HITs are completed, or for other HIT-related events.

Email notifications are useful for small projects, but configuring notifications to use the Amazon Simple Queue Service (SQS) is more reliable for large projects and allows automated processing of notifications.

setnotification() is an alias.

Value

A data frame containing details of the Notification and whether or not the request was successfully executed by MTurk.

Once configured, events will trigger a side effect in the form of a notification sent to the specified transport (either an email address or SQS queue). That notification will contain the following details: EventType, EventTime, HITTypeId, HITId, and (if applicable) AssignmentId.

Note that the 'Notification' column in this dataframe is a dictionary object coerced into a character type. This cannot be used again directly as a notification parameter, but it can be used to re-construct the dictionary object.
Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference: Operation
API Reference: Concept

See Also

GenerateNotification
SendTestEventNotification

Examples

```r
## Not run:
# setup email notification
hittype <- RegisterHITType(title = "10 Question Survey",
                           description = "Complete a 10-question survey",
                           reward = ".20",
                           duration = seconds(hours = 1),
                           keywords = "survey, questionnaire, politics")

a <- GenerateNotification("user@gmail.com", "Email", "AssignmentAccepted")
SetHITTypeNotification(hit.type = hittype$HITTypeId,
                        notification = a,
                        active = TRUE)

# send test notification
SendTestEventNotification(a, test.event.type = "AssignmentAccepted")

## End(Not run)
```

ToDataFrameAssignment  ToDataFrameAssignment

Description

ToDataFrameAssignment

Usage

ToDataFrameAssignment(assignment)

Arguments

assignment  assignment
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Usage</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ToDataFrameBonusPayments</code></td>
<td>ToDoDataFrameBonusPayments</td>
<td><code>ToDataFrameBonusPayments(bonuses)</code></td>
<td><code>bonuses</code></td>
</tr>
<tr>
<td><code>ToDataFrameHITs</code></td>
<td>ToDoDataFrameHITs</td>
<td><code>ToDataFrameHITs(hits)</code></td>
<td><code>hits</code></td>
</tr>
<tr>
<td><code>ToDataFrameQualificationRequests</code></td>
<td>ToDoDataFrameQualificationRequests</td>
<td><code>ToDataFrameQualificationRequests(requests)</code></td>
<td><code>requests</code></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Usage</td>
<td>Arguments</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td><code>ToDataFrameQualificationRequirements</code></td>
<td></td>
<td><code>ToDataFrameQualificationRequirements(hits)</code></td>
<td>hits</td>
</tr>
<tr>
<td><code>ToDataFrameQualifications</code></td>
<td></td>
<td><code>ToDataFrameQualifications(quals)</code></td>
<td>qualifications</td>
</tr>
<tr>
<td><code>ToDataFrameQualificationTypes</code></td>
<td></td>
<td><code>ToDataFrameQualificationTypes(quals)</code></td>
<td>qualifications</td>
</tr>
</tbody>
</table>
ToDataFrameQuestionFormAnswers

Description
ToDataFrameQuestionFormAnswers

Usage
ToDataFrameQuestionFormAnswers(assignment, answers)

Arguments
assignment assignment
answers answers

ToDataFrameReviewableHITs

Description
ToDataFrameReviewableHITs

Usage
ToDataFrameReviewableHITs(hits)

Arguments
hits hits
**ToDataFrameReviewResults**

**Description**

ToDataFrameReviewResults

**Usage**

ToDataFrameReviewResults(results)

**Arguments**

results results

---

**ToDataFrameWorkerBlock**

**Description**

ToDataFrameWorkerBlock

**Usage**

ToDataFrameWorkerBlock(workers)

**Arguments**

workers workers

---

**UpdateQualificationScore**

Update a worker’s score for a QualificationType

**Description**

Update a worker’s score for a QualificationType that you created. Scores for built-in QualificationTypes (e.g., location, worker statistics) cannot be updated.

**Usage**

UpdateQualificationScore(qual, workers, values = NULL, increment = NULL, verbose =getOption("pyMTurkRVerbose", TRUE))
Arguments

qual A character string containing a QualificationTypeId.

workers A character string containing a WorkerId, or a vector of character strings containing multiple WorkerIds.

values A character string containing an integer value to be assigned to the worker, or a vector of character strings containing integer values to be assigned to each worker (and thus must have length equal to the number of workers).

increment An optional character string specifying, in lieu of “values”, the amount that each worker’s current QualificationScore should be increased (or decreased).

verbose Optionally print the results of the API request to the standard output. Default is taken from `getOption('pyMTurkR.verbose', TRUE)`.

Details

A function to update the Qualification score assigned to one or more workers for the specified custom QualificationType. The simplest use is to specify a QualificationTypeId, a WorkerId, and a value to be assigned to the worker. Scores for multiple workers can be updated in one request.

Additionally, the `increment` parameter allows you to increase (or decrease) each of the specified workers scores by the specified amount. This might be useful, for example, to keep a QualificationType that records how many of a specific style of HIT a worker has completed and increase the value of each worker’s score by 1 after they complete a HIT.

This function will only affect workers who already have a score for the QualificationType. If a worker is given who does not already have a score, they will not be modified.

updatequalscore() is an alias.

Value

A data frame containing the QualificationTypeId, WorkerId, Qualification score, and whether the request to update each was valid.

Author(s)

Tyler Burleigh, Thomas J. Leeper

References

API Reference

See Also

GetQualificationScore
GetQualifications
UpdateQualificationType

**Description**

Update characteristics of a QualificationType.

**Usage**

```r
UpdateQualificationType(qual, description = NULL, status = NULL,
retry.delay = NULL, test = NULL, answerkey = NULL,
test.duration = NULL, auto = NULL, auto.value = NULL,
verbose =getOption("pyMTurkR.verbose", TRUE))
```

**Arguments**

- **qual**
  A character string containing a QualificationTypeId.

- **description**
  A longer description of the QualificationType. This is visible to workers. Maximum of 2000 characters.

- **status**
  A character vector of “Active” or “Inactive”, indicating whether the QualificationType should be active and visible.

- **retry.delay**
  An optional time (in seconds) indicating how long workers have to wait before requesting the QualificationType after an initial rejection. If not specified, retries are disabled and Workers can request a Qualification of this type only once, even if the Worker has not been granted the Qualification.

- **test**
  An optional character string consisting of a QuestionForm data structure, used as a test a worker must complete before the QualificationType is granted to them.

- **answerkey**
  An optional character string consisting of an AnswerKey data structure, used to automatically score the test
test.duration  An optional time (in seconds) indicating how long workers have to complete the test.

auto  A logical indicating whether the Qualification is automatically granted to workers who request it. Default is NULL meaning FALSE.

auto.value  An optional parameter specifying the value that is automatically assigned to workers when they request it (if the Qualification is automatically granted).

verbose  Optionally print the results of the API request to the standard output. Default is taken fromgetOption('pyMTurkR.verbose',TRUE).

Details
A function to update the characteristics of a QualificationType. Name and keywords cannot be modified after a QualificationType is created.
updatequal() is an alias.

Value
A data frame containing the QualificationTypeId of the newly created QualificationType and other details as specified in the request.

Author(s)
Tyler Burleigh, Thomas J. Leeper

References
API Reference

See Also
GetQualificationType
CreateQualificationType
DisposeQualificationType
SearchQualificationTypes

Examples

## Not run:
qual1 <- CreateQualificationType(name="Worked for me before",
   description="This qualification is for people who have worked for me before",
   status = "Active",
   keywords="Worked for me before")
qual2 <- UpdateQualificationType(qual1$QualificationTypeId,
   description="This qualification is for everybody!",
   auto=TRUE, auto.value="5")
DisposeQualificationType(qual1$QualificationTypeId)

## End(Not run)
# Index

**Topic Assignments**
- approveAssignment, 5
- GetAssignment, 34
- RejectAssignment, 53

**Topic HITs**
- ChangeHITType, 9
- CreateHIT, 14
- DisableHIT, 18
- ExtendHIT, 22
- GenerateExternalQuestion, 24
- GenerateHITReviewPolicy, 25
- GenerateHITsFromTemplate, 28
- GenerateHTMLQuestion, 30
- GetHIT, 38
- GetHITsForQualificationType, 39
- GetReviewableHITs, 46
- GetReviewResultsForHIT, 47
- RegisterHITType, 51
- SearchHITs, 56
- SearchQualificationTypes, 57
- SetHITAsReviewing, 60

**Topic Notifications**
- GenerateNotification, 32
- SendTestEventNotification, 59
- SetHITTypeNotification, 62

**Topic Qualifications**
- AssignQualification, 6
- CreateQualificationType, 16
- GetHITsForQualificationType, 39
- GetQualificationRequests, 41
- GetQualifications, 42
- GetQualificationScore, 44
- GetQualificationType, 45
- GrantQualification, 50
- RevokeQualification, 54
- UpdateQualificationScore, 67
- UpdateQualificationType, 69

**Topic Workers**
- BlockWorker, 8
- ContactWorker, 12
- GetBonuses, 36
- GrantBonus, 48

**Topic package**
- pyMTurkR-package, 3

AcceptQualificationRequest (GrantQualification), 50
acceptrequest (GrantQualification), 50
AccountBalance, 4
accountbalance (AccountBalance), 4
approve (ApproveAssignment), 5
approveall (ApproveAssignment), 5
ApproveAllAssignments, 54
ApproveAllAssignments (ApproveAssignment), 5
ApproveAssignment, 5, 54
ApproveAssignments, 3
ApproveAssignments (ApproveAssignment), 5
assignment (GetAssignment), 34
assignments (GetAssignment), 34
assignqual (AssignQualification), 6
AssignQualification, 6, 17
AssignQualifications (AssignQualification), 6
AssociateQualificationWithWorker (AssignQualification), 6
block (BlockWorker), 8
blockedworkers (BlockWorker), 8
BlockWorker, 8
BlockWorkers (BlockWorker), 8
bonus (GrantBonus), 48
bonuses (GetBonuses), 36
ChangeHITType, 9, 52, 53
changehittype (ChangeHITType), 9
CheckAWSKeys (GetClient), 37
contact (ContactWorker), 12
ContactWorker, 3, 12
ContactWorkers (ContactWorker), 12
create (CreateHIT), 14
CreateHIT, 3, 11, 14, 17, 24, 25, 30, 31, 33, 48, 53, 58
createhit (CreateHIT), 14
CreateHITType (RegisterHITType), 51
createhittype (RegisterHITType), 51
CreateHITWithHITType (CreateHIT), 14
createhittwifthittype (CreateHIT), 14
createqual (CreateQualificationType), 16
CreateQualificationType, 6, 7, 16, 45, 70
CreateWorkerBlock (BlockWorker), 8
DeleteHIT (DisableHIT), 18
deletehit (DisableHIT), 18
deleteequal (DisposeQualificationType), 20
DeleteQualificationType
 (DisposeQualificationType), 20
DeleteWorkerBlock (BlockWorker), 8
DisableHIT, 18
disablehit (DisableHIT), 18
DisassociateQualificationFromWorker
 (RevokeQualification), 54
DisposeHIT (DisableHIT), 18
disposehit (DisableHIT), 18
disposequal (DisposeQualificationType), 20
DisposeQualificationType, 20, 70
emptydf, 21
ExpireHIT (DisableHIT), 18
extend (ExtendHIT), 22
ExtendHIT, 22
GenerateAssignmentReviewPolicy, 14
GenerateAssignmentReviewPolicy
 (GenerateHITReviewPolicy), 25
GenerateExternalQuestion, 24, 31
GenerateHITReviewPolicy, 14, 25, 48
GenerateHITSFromTemplate, 15, 28
GenerateHTMLQuestion, 30
GenerateNotification, 32, 59, 62, 63
GenerateQualificationRequirement, 10, 15, 33, 52
g_account_balance (AccountBalance), 4
GetAssignment, 5, 34, 54
GetAssignments, 3
GetAssignments (GetAssignment), 34
g_balance (AccountBalance), 4
GetBlockedWorkers (BlockWorker), 8
GetBonuses, 36, 49
gbonuses (GetBonuses), 36
GetClient, 37
GetHIT, 38, 46
gethit (GetHIT), 38
gethitsbyqual
 (GetHITSForQualificationType), 39
GetHITSForQualificationType, 39
GetQualificationRequests, 41, 50, 51
GetQualifications, 42, 44, 68
GetQualificationScore, 43, 44, 68
GetQualificationType, 45, 70
getquals (GetQualifications), 42
GetReviewableHITS, 46, 61
GetReviewResultsForHIT, 47
GrantBonus, 3, 37, 48
grantqual (GrantQualification), 50
GrantQualification, 41, 42, 50, 55
GrantQualifications
 (GrantQualification), 50
hit (GetHIT), 38
HITStatus (GetHIT), 38
hittype (RegisterHITType), 51
ListAssignmentsForHIT (GetAssignment), 34
ListBonusPayments (GetBonuses), 36
listbonuspayments (GetBonuses), 36
ListHITS (SearchHITS), 56
listhits (SearchHITS), 56
ListHITSForQualificationType
 (GetHITSForQualificationType), 39
ListQualificationRequests
 (GetQualificationRequests), 41
ListQualificationTypes
 (SearchQualificationTypes), 57
ListQuals (SearchQualificationTypes), 57
listquals (SearchQualificationTypes), 57
ListReviewPolicyResultsForHIT
 (GetReviewResultsForHIT), 47
ListWorkerBlocks (BlockWorker), 8
INDEX

listworkerblocks (BlockWorker), 8
ListWorkersWithQualificationType (GetQualifications), 42
notificationtest (SendTestEventNotification), 59
notify (ContactWorker), 12
NotifyWorker (ContactWorker), 12
NotifyWorkers (ContactWorker), 12
paybonus (GrantBonus), 48
pyMTurkR (pyMTurkR-package), 3
pyMTurkR-package, 3
qualrequests (GetQualificationRequests), 41
qualscore (GetQualificationScore), 44
qualtype (GetQualificationType), 45
RegisterHITType, 11, 17, 33, 51
reject (RejectAssignment), 53
RejectAssignment, 6, 53
RejectAssignments (RejectAssignment), 53
RejectQualification (GrantQualification), 41, 42, 55
RejectQualification (GrantQualification), 50
RejectQualifications, 51
RejectQualifications (GrantQualification), 50
rejectrequest (GetReviewableHITs), 46
reviewable (GetReviewableHITs), 46
reviewing (SetHITAsReviewing), 60
reviewresults (GetReviewResultsForHIT), 47
revokequal (RevokeQualification), 54
RevokeQualification, 51, 54
RevokeQualifications (RevokeQualification), 54
SearchHITs, 46, 56
searchhits (SearchHITs), 56
SearchQualifications (SearchQualificationTypes), 57
SearchQualificationTypes, 57, 70
SearchQuals (SearchQualificationTypes), 57
searchquals (SearchQualificationTypes), 57
seconds, 10, 15, 22, 52, 58
sendbonus (GrantBonus), 48
SendTestEventNotification, 33, 59, 63
SetHITAsReviewing, 60
SetHITTypeNotification, 32, 33, 60, 62
setnotification (SetHITTypeNotification), 62
StartClient (GetClient), 37
status (GetHIT), 38
ToDataFrameAssignment, 63
ToDataFrameBonusPayments, 64
ToDataFrameHITs, 64
ToDataFrameQualificationRequests, 64
ToDataFrameQualificationRequirements, 65
ToDataFrameQualifications, 65
ToDataFrameQualificationTypes, 65
ToDataFrameQuestionFormAnswers, 66
ToDataFrameReviewableHITs, 66
ToDataFrameReviewResults, 67
ToDataFrameWorkerBlock, 67
unblock (BlockWorker), 8
UnblockWorker (BlockWorker), 8
UnblockWorkers (BlockWorker), 8
UpdateHITReviewStatus (SetHITAsReviewing), 60
UpdateHITTypeOfHIT (ChangeHITType), 9
updatequal (UpdateQualificationType), 69
UpdateQualificationScore, 43, 44, 67
UpdateQualificationType, 16, 17, 69
updatequalscore (UpdateQualificationScore), 67