Amazon Mechanical Turk

Developer Guide

API Version 2011-10-01

Amazon Mechanical Turk: Developer Guide

Welcome 1
Introduction to Amazon Mechanical Turk 2
Making Requests 7
Making SOAP Requests 7
Making REST Requests 9
AWS Request Authentication 11
Understanding Responses 15
Understanding Requesters and Workers 17
Working With HITs 19
Understanding HIT Types 23
Creating and Managing Assignments 25
Creating and Managing Qualifications 31
Creating and Managing Notifications 36
Document History 39

Welcome

This is the *Amazon Mechanical Turk Developer Guide*. This guide provides developers with a conceptual overview of Amazon Mechanical Turk and describes how to programmatically interact with the Mechanical Turk web service.

Amazon Mechanical Turk is a web service that provides an on-demand, scalable, human workforce to complete jobs that humans can do better than computers, for example, recognizing objects in photos. For more information about this product go to Amazon Mechanical Turk.

How Do I...?

How do I...?	Relevant Topics
Get a general product overview of Mechanical Turk	Introduction to Amazon Mechanical Turk (p. 2)
Send requests to and handle responses from the Mechanical Turk web service	Making Requests (p. 7)
Create a Human Intelligence Task (HIT)	Working With HITs (p. 19)
Understand Mechanical Turk Requesters and Workers	Understanding Requesters and Workers (p. 17)
Create and manage assignments	Creating and Managing Assignments (p. 25)
Create and manage qualifications	Creating and Managing Qualifications (p. 31)
Create and manage notifications	Creating and Managing Notifications (p. 36)

Introduction to Amazon Mechanical Turk

This introduction to Amazon Mechanical Turk provides a detailed summary of this web service. After reading this section, you should have a good idea what it offers and how it can fit in with your business.

Overview of Amazon Mechanical Turk

Amazon Mechanical Turk provides a workforce on demand. Based on the concept that people can do some tasks far better that computer, Amazon Mechanical Turk give you a way to post tasks on the Internet for people to tackle. Those tasks might be determining if there is a specific object in a photo, what color dress looks better than another, or reporting on restaurants in an area. The tasks are posted on the Amazon Mechanical Turk web site, workers complete the tasks and send the results back to Amazon Mechanical Turk where you, the requester (the person who created and pays workers for completing the tasks) can evaluate the work done and thereby pay for the work (or not) and pay bonuses (or not).

This overview describes the business model and the major features of Amazon Mechanical Turk.

Business Model

Amazon Mechanical Turk works some ways similar to a job board. Requesters advertise jobs they are willing to pay people to do. Workers look at the jobs available from all of the Requesters and choose to work on the jobs that interest them and the ones they qualify for. Requesters review submitted work either manually or programmatically and agree to pay for the work or not.

What makes Amazon Mechanical Turk special is that the job advertising and job completion happens over the Internet so the workforce is international and numbers in the hundreds of thousands. The workforce scales with the Requester's needs from none to hundreds, as specified by the Requester.

Advantages

Following are the major advantages of Amazon Mechanical Turk.

- **On demand workforce—**Post jobs to a worldwide set of Workers only when your business needs the help
  Your obligation to those Workers ends when they complete their work.

- **Scalable workforce—**You can use a few or thousands of Workers to complete your jobs
  You can limit the amount of work each Worker can do for you.
- **Qualified workforce—**You can give potential Workers qualification tests
  When your jobs require specialized knowledge or skills, you can create (or use a standardized) qualification test to make sure the Workers performing the job have the skills to complete the job successfully.
- **Pay only for satisfactory work—**You can reject inferior work
  To pay Workers for the work they've done, you have to accept their work. Rejecting their work means they do not get paid. You can even choose to block Workers from working on your jobs.
- **Various user interfaces—**Amazon Mechanical Turk offers a command line interface (CLI), API, and the Requester User Interface
  The CLI gives you hands-on control of Amazon Mechanical Turk functionality. The API enables you to use Amazon Mechanical Turk functionality programmatically. The Requester User Interface enables you to publish a large number of (closely related) jobs with minimal effort.

Amazon Mechanical Turk Concepts

This section describes the concepts and terminology you need to understand to use Amazon Mechanical Turk effectively. They are presented in the order you will most like encounter them.

Requesters

A Requester is a person (or company or organization) who asks questions to Amazon Mechanical Turk. As a Requester, you use a software application to interact with the Amazon Mechanical Turk Service to submit questions, retrieve answers, and perform other automated tasks. You can use the Requester Console (http://requester.mturk.amazon.com/) to check the status of your questions, and manage your account.

To Workers, you are known as the creator of your HITs, and as the creator and maintainer of your Qualification types. Workers see your name, as specified with your Amazon.com account, on the Amazon Mechanical Turk web site.

You perform actions with the Amazon Mechanical Turk Service by using an AWS Access Key ID and AWS Secret Key to cryptographically sign each request. To obtain an AWS Access Key ID and AWS Secret Key, go to http://aws.amazon.com/mturk and sign in with your Amazon.com account e-mail address and password.

Workers

A *Worker* is a person who answers questions for Amazon Mechanical Turk. A Worker uses the Amazon Mechanical Turk web site (http://mturk.amazon.com/) to find questions, submit answers, and manage his or her account.

To Requesters, a Worker is known as the submitter of a HIT assignment, and as a user requesting a Qualification. You see the Worker's account ID (an alphanumeric string assigned by the system) included with assignment data and Qualification requests.

Qualifications represent the Worker's reputation and abilities. A Worker's Qualifications are matched against a HIT's Qualification requirements to allow or disallow the Worker to accept the HIT. A Worker's Qualifications cannot be accessed directly by other users.

Human Intelligence Tasks (HITs)

Each question your application asks is a *Human Intelligence Task*, or *HIT*. A HIT contains all of the information a Worker needs to answer the question, including information about how the question is shown to the Worker and what kinds of answers would be considered valid.

Each HIT has a *reward*, an amount of money you pay to the Worker that successfully completes the HIT.

You can request that more than one Worker ought to complete a HIT by specifying a `MaxAssignments` property for the HIT. For more information, see Creating and Managing Assignments (p. 25).

Assignments

When a Worker finds a HIT to complete, the Worker *accepts* the HIT. Amazon Mechanical Turk creates an *assignment* to track the completion of the task and store the answer the Worker submits.

Amazon Mechanical Turk reserves the assignment while the Worker is actively working on it, so no other Worker can accept it or submit results. If the Worker fails to complete the assignment before the deadline you specified (the Worker *abandons* the HIT), or if the Worker chooses not to complete it after accepting it (the Worker *returns* the HIT), the assignment is once again made available for other Workers to accept.

A HIT can have multiple assignments. This is useful for gathering multiple answers to a single question for comparison, or for collecting multiple opinions. A Worker can only accept a HIT once, so a HIT with multiple assignments is guaranteed to be performed by multiple Workers.

You can specify the maximum number of assignments that any Worker can accept for your HITs. You can set two types of limits:

- The maximum number of assignments any Worker can accept for a specific HIT type you've created
- The maximum number of assignments any Worker can accept for all your HITs that don't otherwise have a HIT-type-specific limit already assigned

For more information, see Creating and Managing Assignments (p. 25).

Approval and Payment

Once a HIT has all of the answers that were requested, or an expiration date you specified has passed, your application retrieves the assignments with the answer data. If an assignment's answer satisfies the question, you *approve* the assignment. You may *reject* the assignment if the HIT was not completed successfully.

Amazon Mechanical Turk automatically processes payment of the reward to the Worker once the assignment is approved. The reward is transferred from your Amazon.com account to the Worker's Amazon.com account. You can deposit or withdraw funds from your Amazon Mechanical Turk account at any time using the Requester web site (http://requester.mturk.amazon.com/).

Qualifications and Quality Control

You can manage which Workers can accept a particular HIT using *Qualifications*. A Qualification is an attribute assigned by you to a Worker. It includes a name and a number value. A HIT can include *Qualification requirements* that a Worker must meet before they are allowed to accept the HIT. Each `QualificationRequirement` describes an expression that a score or metric about the Worker must match for the Worker to be considered "qualified" to complete the HIT. For more information, see Creating and Managing Qualifications (p. 31).

You create a *Qualification type* to represent a Worker's skill or ability. A Worker discovers your Qualification type either by browsing HITs that require it, or by browsing Qualification types directly. The Worker requests a Qualification of the type, and you grant the request with a value.

A Qualification type may include a *Qualification test*. A Qualification test is a set of questions, similar to a HIT, that the Worker must answer to request the Qualification. You can grant the request manually by evaluating the Worker's test answers, or you can include an *answer key* for the test when they create the Qualification type. For Qualification types with a test and an answer key, Amazon Mechanical Turk processes Qualification requests automatically, and sets Qualification values as specified by the answer key.

Amazon Mechanical Turk provides several *system Qualifications* that represent a Worker's account history. The values are updated continuously as the Worker uses the system. A HIT may include Qualification requirements based on these system Qualifications.

For more information, see Creating and Managing Qualifications (p. 31).

Questions and Answers

The `Question` field of a HIT describes what is being asked of the Worker. It includes any information required to answer the question, such as text or images, as well as a description of the range possible answers.

Tip

The Amazon Mechanical Turk Service passes questions and answers between your Requester application and Workers. Workers read questions and provide answers using the Amazon Mechanical Turk web site. The format of this data is device-independent, so future Worker interfaces to Amazon Mechanical Turk can be built on platforms with varying capabilities.

Be aware that the Worker interface is not guaranteed to display your questions in a particular way, nor is it guaranteed to return answers within the ranges you specify in the question form. Amazon Mechanical Turk only ensures that the question and answer data conform to the appropriate schemas.

You can include several different kinds of data in a HIT question:

- Simple text elements, such as paragraphs, headings, and bulleted lists
- Blocks of formatted content that contain XHTML markup, such as for tables, formatted text (bold, italic), and other XHTML features
- Links to images, audio and video, which are typically displayed embedded in the HIT display
- Links to Java applets and Flash movies (which can be interactive), displayed embedded in the HIT display

The question form specification may include multiple fields, or "questions." A question can have the Worker select zero, one or more options from a list (true/false, multiple choice), or it can have the Worker type in text or a number. A field can also request that the Worker upload a file.

The question form specification may suggest the style of a field, guiding how a question may appear to the Worker. Multiple choice questions may appear as radio buttons, checkboxes, or a dropdown list, among others. The suggested style is not guaranteed, since Amazon Mechanical Turk may adjust the appearance to fit the device the Worker is using to see the question.

The specification may also suggest ranges of possible answers for the question. It is up to the device presenting the question to the Worker to validate the Worker's answers, so the results in the assignment are not guaranteed to meet these criteria. Your application should always validate the answers it receives.

Tip

For more information about the question and answer specification format, see QuestionForm Data Structure.

Architectural Overview of Amazon Mechanical Turk

Three types of people interact with Amazon Mechanical Turk:

- Requesters, who creates and pays for the work done by Workers
  Requesters advertise work online through Amazon Mechanical Turk.
- Workers, who find and accept work advertised by Requesters
- Developers, who create Amazon Mechanical Turk applications that Requesters and Workers use
  Requesters can create and advertise work using the Amazon Mechanical Turk command line interface or the Requester User Interface and thereby not need developers

The following table describes a typical Amazon Mechanical Turk workflow.

Amazon Mechanical Turk Workflow

1	A developer creates an Amazon Mechanical Turk application.
2	A Requester uses a Amazon Mechanical Turk application, command line interface, or the Requester UI to create work, called a HIT, and advertises the work using Amazon Mechanical Turk.
3	Workers visit the Amazon Mechanical Turk web site and decide which work to undertake.
4	Optionally, the Requester can require the Worker to pass a qualification test before being granted the opportunity to do the work.
5	The Workers complete one or more HITs and submit their answers using Amazon Mechanical Turk.
6	The Requester reviews the work and pays the Worker for work done well or rejects the work and does not pay the Worker.

Making Requests

Topics

- Making SOAP Requests (p. 7)
- Making REST Requests (p. 9)
- AWS Request Authentication (p. 11)
- Understanding Responses (p. 15)

This section describes how to interact with the Amazon Mechanical Turk Service, how to authenticate and send requests, and how to understand responses.

Making SOAP Requests

This article explains how to make a SOAP request to the Amazon Mechanical Turk web service.

Using SOAP

The Amazon Mechanical Turk web service supports the SOAP message protocol for calling service operations over an HTTP connection. The easiest way to use the SOAP interface with your application is to use a SOAP toolkit appropriate for your platform. SOAP toolkits are available for most popular programming languages and platforms.

The service's Web Services Definition Language (WSDL) file describes the operations and the format and data types of their requests and responses. Your SOAP toolkit interprets the WSDL file to provide your application access to the operations. For most toolkits, your application calls a service operation using routines and classes provided or generated by the toolkit.

The location of the WSDL file is discussed in the section, WSDL Location.

Note

Amazon Mechanical Turk limits the velocity of requests. If you exceed the limit you will receive a 500 or 503 Service Unavailable error. It is highly unlikely that you will reach this limit with normal activity.

Using Operation Parameters With SOAP

The API reference in this guide describes the parameters for each operation and their values. You may find it useful to refer to the WSDL file directly to see how the parameters will appear in the XML of the request generated by your toolkit, and understand how your toolkit will make the operations available to your application code.

The Structure of a Request Message

A SOAP request is an XML data structure generated by your SOAP toolkit and sent to the service. As described by the service WSDL, the root element of this structure is named after the operation, and contains the parameter data for the request.

The root element contains the *AWSAccessKeyId*, *Timestamp*, and *Signature* used to authenticate the request as being sent by you. For more information on these values, see AWS Request Authentication (p. 11).

In addition to the request authentication parameters, the root element contains a *Request* element, which contains the parameters of the specific operation being called. For a description of an operation's parameters, see the appropriate page for the operation in the Amazon Mechanical Turk API Reference . The *Request* element may also contain a *ResponseGroup* parameter, which controls how much data is returned by the service for an operation.

For more information about these parameters and their values, see Common Parameters .

The XML Message for a GetHIT SOAP Request

The following example is the XML for a SOAP message that calls the GetHIT operation. While you will probably not be building the SOAP message for a service request manually, it is useful to see what your SOAP toolkit will try to produce when provided with the appropriate values. Many SOAP toolkits require that you build a request data structure similar to the XML to make a request.

The *GetHIT* element contains the parameters common to all requests, and a *Request* element that contains the operation-specific *HITId* parameter, along with the *ResponseGroup*.

The following example calls the GetHIT operation.

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope
     xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <GetHIT
         xmlns="http://mechanicalturk.amazonaws.com/AWSMechanicalTurkRe
quester/2011-10-01">
      <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
      <Timestamp>2005-10-10T00:00:00.000Z</Timestamp>
      <Signature>[...]</Signature>
      <Request>
        <HITId>123RVWYBAZW00EXAMPLE</HITId>
        <ResponseGroup>Minimal</ResponseGroup>
        <ResponseGroup>HITDetail</ResponseGroup>
      </Request>
    </GetHIT>
```

```
  </soapenv:Body>
</soapenv:Envelope>
```

Making REST Requests

This article explains how to make a REST request to the Amazon Mechanical Turk web service.

Using REST

The Amazon Mechanical Turk web service supports REST requests for calling service operations. REST requests are simple HTTP requests, using either the GET method with parameters in the URL, or the POST method with parameters in the POST body. The response is an XML document that conforms to a schema. You might use REST requests when a SOAP toolkit is not available for your platform, or if REST requests would be easier to make than a heavier SOAP equivalent.

The location of the schema that describes the responses for the various operations is discussed in the section, WSDL Location.

Note

Amazon Mechanical Turk limits the velocity of requests. If you exceed the limit you will receive a 503 Service Unavailable error. It is highly unlikely that you will reach this limit with normal activity.

Using Operation Parameters With REST

The API reference in this guide lists the parameters for each operation. Most parameters can be specified in a REST request using just the name of the parameter and an appropriate value, with the value URL-encoded as necessary to make the request a valid URL.

Some parameters have multiple components. For example, a HIT reward is specified as a *Reward* parameter, which includes an *Amount* and a *CurrencyCode*. In a SOAP request or in a response, this value would appear as an XML data structure. The following code sample demonstrates this data structure.

```
<Reward>
  <Amount>32</Amount>
  <CurrencyCode>USD</CurrencyCode>
</Reward>
```

In a REST request, the components are specified as separate parameters. The name of each component parameter is the main parameter name (such as "Reward"), a dot, a sequence number, a dot, and the component name (such as "Amount"). For example, the preceding example would appear in a REST request as follows:

```
http://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
[...]
&Reward.1.Amount=32
&Reward.1.CurrencyCode=USD
```

For parameters that can be specified more than once in a single request, each parameter name includes a number to make it clear which values belong to each parameter. Parameters with single values simply use the name, a dot, and a number. This is shown in the following example.

```
...&ParameterName.1=valueOne&ParameterName.2=valueTwo...
```

Parameters with component values use the name of the main parameter, followed by a dot, the number, a dot, and the component name. For example, a request for the `CreateHIT` operation can specify more than one *`QualificationRequirement`* for the HIT being created. The value of the *`QualificationRequirement`* parameter is a structure with three components, *`QualificationTypeId`*, *`Comparator`*, and *`Value`*. For example, in an XML message, this structure looks like this:

```
<QualificationRequirement>
  <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
  <Comparator>GreaterThan</Comparator>
  <Value>18</Value>
</QualificationRequirement>
```

The following example shows how a single *`QualificationRequirement`* is specified in a REST request.

```
http://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
[...]
&QualificationRequirement.1.QualificationTypeId=789RVWYBAZW00EXAMPLE
&QualificationRequirement.1.Comparator=GreaterThan
&QualificationRequirement.1.Value=18
```

The following example shows how multiple *`QualificationRequirement`* values are specified.

```
http://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
[...]
&QualificationRequirement.1.QualificationTypeId=789RVWYBAZW00EXAMPLE
&QualificationRequirement.1.Comparator=GreaterThan
&QualificationRequirement.1.Value=18
&QualificationRequirement.2.QualificationTypeId=231FOOYBARW00EXAMPLE
&QualificationRequirement.2.Comparator=GreaterThan
&QualificationRequirement.2.Value=75
```

Parameters Specific to REST Requests

In addition to the parameters found in request data structures, REST requests have additional parameters to indicate the name of the service and the version of the API. (SOAP requests have this information embedded in the SOAP URL.) For more information about these parameters and their values, see Common Parameters . For more information about service versions, see WSDL Location.

Sample REST Request

The following example is a REST request (GET method) that calls the `GetAccountBalance` operation.

```
http://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2011-10-01
```

```
&Operation=GetAccountBalance
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&ResponseGroup.0=Minimal
&ResponseGroup.1=Request
```

AWS Request Authentication

Topics

- AWS Accounts (p. 11)
- Authenticating Requests (p. 12)
- Summary of AWS Request Authentication (p. 12)
- Calculating Request Signatures (p. 12)
- Using REST and SOAP Transactions (p. 13)
- URL Encoding (p. 13)
- Code Samples for Request Authentication (p. 13)

Request authentication is the process of verifying the identity of the sender of a request. In the context of Amazon Web Services (AWS) requests, authentication is the process by which AWS can confirm that a request came from a registered user, as well as the identity of that registered user.

To enable authentication, each request must carry information about the identity of the request sender. The request must also contain additional information that AWS can use to verify that the request can only have been produced by the sender identified. If the request passes this verification test it is determined to be “authentic” and AWS has sufficient information to verify the identity of the sender.

Verifying the identity of the sender of a request is important, as it ensures that only those requests made by the person or party responsible for the AWS account specified in the request are accepted and allowed to interact with AWS services. In this manner, request authentication allows Amazon to track the usage of AWS services on a per request basis. This enables Amazon to charge and bill AWS subscribers for use of AWS paid (not free) services.

AWS Accounts

To access Amazon web services, a developer must create an AWS account. AWS accounts are associated with Amazon.com accounts. To sign in to an AWS account, a developer uses his or her Amazon.com account e-mail and password.

Upon creating the AWS account, the developer is assigned an Access Key ID (*AWSAccessKeyId*) and a Secret Access Key. The Access Key ID, which is associated with the AWS account, is used in requests to identify the party responsible for the request. However, because an Access Key ID is sent as a request parameter, it is not secret and could be used by anyone sending a request to AWS. To protect from impersonation, the request sender must provide additional information that can be used to verify the sender’s identity and ensure that the request is legitimate. This additional information, a request signature that is calculated using the Secret Access Key, demonstrates possession of a shared secret known only to AWS and the sender of the request. A Secret Access Key is a 20-character alphanumeric sequence generated by AWS.

Authenticating Requests

Requests to AWS are authenticated by verifying information contained within the request. This verification is performed using the information in the following table.

Parameter	Description
AWSAccessKeyId	The sender's AWS account is identified by the Access Key ID. The Access Key ID is used to look up the Secret Access Key.
Signature	Each request to a web service that requires authenticated requests must contain a valid request signature, or the request is rejected. A request signature is calculated using the Secret Access Key assigned to the developer's account by AWS, which is a shared secret known only to AWS and the developer.
Timestamp	The date and time the request was created, represented as a string in UTC. The format of the value of this parameter must match the format of the XML Schema dateTime data type.

Summary of AWS Request Authentication

Following are the basic tasks used in authenticating requests to AWS. It is assumed that the developer has already registered with AWS and received an Access Key ID and Secret Access Key.

Authenticating Requests

1	The sender constructs a request to AWS.
2	The sender calculates a Keyed-Hashing for Message Authentication code (HMAC), the request signature using the his or her Secret Access Key and the values of the *Service*, *Operation*, and *Timestamp* parameters as input.
3	The sender of the request sends the request data, the signature, and Access Key ID (the key-identifier of the Secret Access Key used) to AWS.
4	AWS uses the Access Key ID to look up the Secret Access Key.
5	AWS generates a signature from the request data and the Secret Access Key using the same algorithm used to calculate the signature in the request.
6	If the signature generated by AWS matches the one sent in the request, the request is considered to be authentic. If the comparison fails, the request is discarded, and AWS returns an error response.

Calculating Request Signatures

A request signature, an HMAC, is calculated by concatenating the values of the *Service*, *Operation*, and *Timestamp* parameters, in that order, and then calculating an RFC 2104-compliant HMAC, using the Secret Access Key as the "key." The computed HMAC value should be base64 encoded, and is passed as the value of the *Signature* request parameter. For more information, go to http://www.faqs.org/rfcs/rfc2104.html.

When a request is received, AWS verifies the request signature by computing an HMAC value for the request and comparing the value of that HMAC with the value in the request. If the computed HMAC

value matches the HMAC value in the request, the identity of the sender is verified and the request is accepted. If the values do not match the request is rejected, and an error is returned.

Using REST and SOAP Transactions

Requests can be sent using REST (XML over HTTP) or SOAP. The contents of the request are the same, only the request format differs.

URL Encoding

The result of the SHA-1 hash is binary data. An encoding must be specified to include this in either a SOAP or REST request. Both REST and SOAP requests should be Base64 encoded.

However, as the results of Base64 encoding can contain characters that are not legal in a URL, such as plus signs (+),slashes (/), and equal signs (=), results for REST requests should be URL encoded, as specified in RFC 1738, section 2.2.

Code Samples for Request Authentication

Calculating an HMAC Request Signature

The following code sample demonstrates how to calculate a request signature to sign authenticated requests to AWS.

```
package amazon.webservices.common;

import java.security.SignatureException;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

/**
 * This class defines common routines for generating
 * authentication signatures for AWS Platform requests.
 */

public class Signature {

    private static final String HMAC_SHA1_ALGORITHM = "HmacSHA1";

    /**
     * Computes RFC 2104-compliant HMAC signature.
     *
     * @param data
     *     The data to be signed.
     * @param key
     *     The signing key.
     * @return
     *     The Base64-encoded RFC 2104-compliant HMAC signature.
     * @throws
     *     java.security.SignatureException when signature generation fails
     */
    public static String calculateRFC2104HMAC(String data, String key)
        throws java.security.SignatureException
```

```
    {
        String result;
        try {
            // get an hmac_sha1 key from the raw key bytes
            SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(),
                                                         HMAC_SHA1_ALGORITHM);


            // get an hmac_sha1 Mac instance and initialize with the signing
key
            Mac mac = Mac.getInstance(HMAC_SHA1_ALGORITHM);
            mac.init(signingKey);

            // compute the hmac on input data bytes
            byte[] rawHmac = mac.doFinal(data.getBytes());

            // base64-encode the hmac
            result = Encoding.EncodeBase64(rawHmac);
        }
        catch (Exception e) {
            throw new SignatureException("Failed to generate HMAC : " + e.get
Message());
        }
        return result;
    }
}
```

Data Encoding

This sample, provided in support of the previous sample for calculating HMAC signatures, demonstrates how to perform Base64 encoding of input types in AWS requests.

```
package amazon.webservices.common;

/**
 * This class defines common routines for encoding
 * data in AWS Platform requests.
 */

public class Encoding {

    /**
     * Performs base64-encoding of input bytes.
     *
     * @param rawData
     *      Array of bytes to be encoded.
     * @return
     *      The base64-encoded string representation of rawData.
     */
    public static String EncodeBase64(byte[] rawData) {
        return Base64.encodeBytes(rawData);
    }
}
```

Performing Base64 Encoding and Decoding

This sample demonstrates how to encode and decode to and from Base64 notation. The code for this sample is not included in this document due to the length of the file. For a complete copy of the code, go to http://iharder.net/base64.

Understanding Responses

This article describes the structure of responses from the Amazon Mechanical Turk web service.

Response Messages, SOAP and REST

In response to an operation call, the Amazon Mechanical Turk web service returns an XML data structure that contains the results of the call. This data conforms to a schema.

For SOAP requests, this data structure is the SOAP message body of the response. SOAP toolkits typically convert the response data into structures for use with your programming language, or allow you to specify your own data bindings.

For REST requests, this data structure is simply the body of the HTTP response. You can use a data binding method for REST responses, or use an XML parser directly to process the information.

Other than the use of a message envelope in the case of SOAP, the schema for the results is the same for both SOAP and REST responses. The SOAP WSDL imports an XSD file to define the response messages, and REST users can access the XSD file directly. For more information, see WSDL Location.

The Structure of a Response

The response message is returned in an XML element named after the operation. For example, the `GetHIT` operation returns a response element named *`GetHITResponse`*.

This element contains an *`OperationRequest`* element, and a "result" element.

OperationRequest

The *`OperationRequest`* element contains information about the request. It always contains a *`RequestId`* element, a unique identifier assigned by the service to this specific operation call.

If an operation call is unsuccessful, *`OperationRequest`* contains an *`Errors`* element, with one or more *`Error`* elements. Each *`Error`* includes:

- A *`Code`* that identifies the type of error the occurred,
- A *`Message`* that describes the error condition in a human-readable form, and
- Zero or more *`Data`* elements that provide information about the error in a machine-readable form. Each *`Data`* has a *`Key`* and a *`Value`*.

If the **`Request`** response group is specified in the request, *`OperationRequest`* includes an *`Arguments`* element that lists all of the parameters that were sent to the operation. It contains one or more *`Arugment`* elements, each with a *`Name`* and a *`Value`*. For more information about response groups, see Common Parameters .

The Result Element

The response message always includes a result element, which contains the result data for the operation call. The name and contents of this element depend on the operation being called. Results are described for each operation in the Amazon Mechanical Turk API Reference .

The result element also contains an *IsValid* element, with a Boolean value indicating if the request was valid. If this value is **false**, the result element usually does not contain anything else.

If the **Request** response group is specified in the request, the result element includes the contents of the request that correspond with the results in the result element. The name of this element depends on the operation. For example, a call to the GetHIT operation that includes the **Request** response group will include a *GetHITRequest* element along with the results. For more information about response groups, see Common Parameters .

A Sample Response Message

The following is an example of a response message that could be returned by a call to the GetHIT operation. For a SOAP request, the message is returned as the response message body, inside a SOAP envelope. For a REST request, the message is returned directly as the body of the HTTP response.

```
<GetHITResponse>
  <OperationRequest>
    <RequestId>XA5TETQ3G6QF7EXAMPLE</RequestId>
  </OperationRequest>
  <HIT>
    <Request>
      <IsValid>true</IsValid>
    </Request>
    <HITId>123RVWYBAZW00EXAMPLE</HITId>
    <CreationTime>2005-10-10T23:59:59.99Z</CreationTime>
    [... other fields of the HIT ...]
  </HIT>
</GetHITResponse>
```

Understanding Requesters and Workers

Working With Amazon Mechanical Turk Accounts

Requesters and Workers are Amazon Mechanical Turk users and have Amazon.com accounts. Account information is managed by Amazon.com, so anyone with an Amazon.com account can use that account's e-mail address and password to sign in to Amazon Mechanical Turk.

Your Amazon.com account holds the money you will pay to Workers as rewards for completing HITs, as well as the money to pay for Amazon Mechanical Turk listing fees. A Worker's Amazon.com account holds the money the Worker receives from Requesters for completing HITs. You can transfer money to and from your Amazon.com account at any time using the Requester Console. Workers transfer money using the Amazon Mechanical Turk web site.

To retrieve your account balance

- Use GetAccountBalance in a request similar to the following.

```
http://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2008-04-01
&Operation=GetAccountBalance
&Signature=[signature for this request]
&Timestamp=[your system's local time]
```

This request retrieves the Requester's account balance.

Using Statistics and System Qualifications

Amazon Mechanical Turk keeps statistics about every user's activity in the system. Workers can view their own statistics using the Amazon Mechanical Turk web site. You can view your own statistics using the Requester Console.

HITs can use some Worker statistics as the basis for Qualification requirements. These are known as system Qualifications.

To get requester statistics

- Use GetRequesterStatistic in a request similar to the following.

```
http://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2008-08-02
&Operation=GetRequesterStatistic
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&Statistic=TotalRewardPayout
&TimePeriod=ThirtyDays
&Count=1
```

This request retrieves the total reward payout for the thirty days leading up to the current date.

Contacting Workers

NotifyWorkers lets you send e-mail to Workers who have interacted with you in the past. Using the Worker ID included with the data the Worker submits to you, you can send a Worker a message without having to know his or her name or e-mail address.

If you have work in the system, Workers can contact you using the Amazon Mechanical Turk web site. Amazon Mechanical Turk relays the message to you by e-mail.

To e-mail Workers you've worked with

- Use NotifyWorkers in a request similar to the following.

```
http://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2008-04-01
&Operation=NotifyWorkers
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&Subject=Thank%20you
&MessageText=Hello!%20Just%20wanted%20to%20say%20thank%20you...
&WorkerId.1=AZ3123EXAMPLE
&WorkerId.2=AZ3456EXAMPLE
&WorkerId.3=AZ3789EXAMPLE
```

This request sends an e-mail message to three Workers.

Working With HITs

Topics

- Creating HITs (p. 19)
- The Title, Description, and Keywords (p. 20)
- The Reward (p. 20)
- Deadlines and Expirations (p. 20)
- Asking Workers to Upload Files (p. 21)
- Using Your Web Site To Host Questions (p. 21)
- The Requester Annotation (p. 22)

Creating HITs

A Human Intelligence Task, or HIT, is a question your application asks, and a Worker answers. Your application submits a HIT using the CreateHITOperation operation.

A HIT includes:

- A title
- A description
- Keywords, used to help Workers find the HITs with a search
- The amount of the reward
- An amount of time in which the Worker must complete the HIT
- An amount of time after which the HIT will no longer be available to Workers
- The number of Workers needed to submit results for the HIT before the HIT is considered complete
- Qualification requirements
- All of the information required to answer the question

Once created, the HIT becomes browsable and searchable on the Amazon Mechanical Turk web site, and can be accepted and completed by a Worker whose Qualifications match the HIT's Qualification requirements.

Note

HITs, Hit types, and Qualifications types are available until they are explicitly disposed (using either the `DisposeHIT` or `DisposeQualificationType` operations), or until they have been inactive for 120 days. HITs and their related assignment data are disposed 120 days after you approve or reject the assignments. HIT types and Qualification types are disposed 120 days after their last use. If you require access to HIT data for longer than 120 days, we recommend you keep a local copy.

The Title, Description, and Keywords

A HIT has a *`Title`*, a *`Description`*, and *`Keywords`* that generally describe the HIT. These values show at a glance what kind of work is involved in the HIT.

The Amazon Mechanical Turk web site includes a search engine for performing keyword searches for HITs. Search terms can match against a HIT's title, description or keywords.

Using International Characters

If a HIT uses international characters in the HIT title, description, or if the results contain international characters, you have to configure Excel to display the international characters.

The Reward

A HIT can have a *`Reward`*, an amount of money paid to the Worker once you approve the results the Worker submitted. This amount is transferred from your account to the Worker's account automatically once you approve the results.

The question is how much should you offer as an award? If you offer to high a reward, you will find that your results will be costly. If you set the reward too low, you run the danger of no one working on your HITs. To set your reward, look for HITs that are similar in terms of the amount and kind of work required of the Worker.

Tip

A minimum HIT listing fee applies to each HIT, even if the reward is zero.

Deadlines and Expirations

Once a Worker has chosen a HIT to work on, Amazon Mechanical Turk starts a timer to keep track of how long the Worker has been holding on to the HIT. If the amount of time exceeds the *`AssignmentDurationInSeconds`* of the HIT, Amazon Mechanical Turk declares that the Worker has "abandoned" the HIT, cancels the Worker's assignment, and makes the HIT available for other Workers to accept. When a new Worker chooses the HIT, the timer starts over.

Amazon Mechanical Turk also keeps track of how long the HIT has been in the system, from the moment the HIT is created. If the HIT's lifetime exceeds the *`LifetimeInSeconds`*, the HIT is declared completed, whether or not all of the requested answers have been submitted by Workers. The HIT is removed from the Amazon Mechanical Turk web site, and is no longer available for Workers to find and complete. For more information about the life cycle of a HIT, see Creating and Managing Assignments (p. 25).

Asking Workers to Upload Files

One type of HIT question can prompt the Worker to upload a file. You specify minimum and maximum sizes for the file, and Amazon Mechanical Turk will ensure that the Worker uploads a file within the specified size range.

The results for the HIT will include the actual size of the file the Worker uploaded. When your application is ready to retrieve the file, it calls GetFileUploadURL with the assignment ID and the question identifier. The operation returns a temporary URL that your application can use to download the file. The URL will only work for 60 seconds after `GetFileUploadURL` is called.

Tip

The 60-second expiration of the temporary URL returned by the `GetFileUploadURL` operation ensures that only your application can access the data, while allowing your application to retrieve the file using a direct HTTP connection to the URL. The time limit only applies to initiating the download; the download itself will take as long as is necessary to retrieve the complete file. If you need to initiate the download after the URL has expired, you can call `GetFileUploadURL` at any time to get a new temporary URL.

Using Your Web Site To Host Questions

If you need more control over the display or logic of how HITs are presented to users than is provided by the Amazon Mechanical Turk content types, you can create a HIT whose question is hosted on your own web site.

An "external question" HIT appears to the Worker as a HIT whose question form is a web browser frame. The Worker's browser loads the contents of the frame directly from a URL you provide when you create the HIT. This gives you complete control over what appears in the frame, and how the Worker interacts with the question.

The Worker submits results for your HIT using the form on your web site. Your form then submits this data back to Amazon Amazon Mechanical Turk, where it is stored with the HIT results. Amazon Mechanical Turk then advances the user's browser to another HIT. The result is similar to submitting any other kind of HIT.

Note

Setting up a HIT with an external question requires a web server capable of functioning under very high load. Similar to images, Java applets and Flash applications, failure to serve files may prevent the Worker from seeing the data required to complete the task. However, unlike images and applets, if the web server fails to function for an external HIT, the Worker may not be able to submit results to Amazon Amazon Mechanical Turk at all.

An external HIT can be as simple as a web form in an HTML file. As such, the web server hosting the external HIT content does not need sophisticated server functionality. For example, you can create a sophisticated web form for an external HIT using HTML and JavaScript, then host the HTML file at Amazon S3.

For information about created HITs with external questions, see ExternalQuestion.

The Requester Annotation

Your application can include a *RequesterAnnotation* for each HIT, a value visible only to you. You can use this value to associate the HIT data with an identifier internal to your application. The Requester annotation is returned with other HIT data, such as from a call to GetHIT.

Understanding HIT Types

HIT Types

In many common uses of Amazon Mechanical Turk, you will want to ask many questions of the same kind, such as identifying an object in each of thousands of photos. A single Worker can answer one or many of these questions. To make it easy for Workers to find your HITs, Amazon Mechanical Turk groups similar HITs together, and Workers search and browse the groups. If a Worker qualifies for HITs in the group, the Worker may preview a single HIT, then accept it and begin work. When a Worker completes one HIT in a group, Amazon Mechanical Turk shows the Worker another HIT from the same group. Most Workers will complete many similar HITs in a single session.

Amazon Mechanical Turk groups HITs together based on their *HIT type*. A HIT type is defined by the values for a set of common properties. Two HITs with identical values for these properties are considered to be of the same HIT type, and appear in the same group on the Amazon Mechanical Turk web site.

Your application can use HIT types to manage different kinds of work. Each HIT type has a HIT type ID, which your application can use to query for HITs of a particular type. The HIT type ID may also be used with the CreateHITOperation operation in place of the common property values, to ensure that the new HIT will be assigned the same type as other HITs.

Note

HITs, Hit types, and Qualifications types are available until they are explicitly disposed (using either the `DisposeHIT` or `DisposeQualificationType` operations), or until they have been inactive for 120 days. HITs and their related assignment data are disposed 120 days after you approve or reject the assignments. HIT types and Qualification types are disposed 120 days after their last use. If you require access to HIT data for longer than 120 days, we recommend you keep a local copy.

Properties of a HIT Type

The HIT properties that define a HIT type are the following:

- Title
- Description
- Keywords

- Reward
- AssignmentDurationInSeconds
- AutoApprovalDelayInSeconds
- A set of zero or more QualificationRequirements

How HIT Types Are Created

You can explicitly register a new HIT type by calling RegisterHITType. This operation takes values for the common parameters, and returns a HIT type ID for the type that matches the values. You can create HITs using the new type by calling the CreateHITOperation operation with the type ID, and values for the properties specific to the HIT (such as the *`Question`*).

You can create a HIT without specifying a HIT type ID by passing values for the common parameters directly to the call to CreateHITOperation . If the values match an existing HIT type, the HIT will be given the existing type. If the values do not match an existing HIT type, a new type is created with those values. In either case, the HIT type ID is returned with the HIT data in the response from the call to CreateHITOperation .

Tip

To avoid accidentally passing mismatched values for two HITs that ought to be part of the same type, register the HIT type using RegisterHITType, then create the HITs using the HIT type ID.

How to Change the HIT Type

Once you have created a HIT, you might want to change the HIT type. Use the ChangeHITTypeOfHITOperation operation to do this.

Properties Specific to a HIT

HITs of the same HIT type can have differing values for the following properties:

- Question
- MaxAssignments
- LifetimeInSeconds
- RequesterAnnotation

Creating and Managing Assignments

Topics

- A Worker Accepts a HIT (p. 25)
- Multiple Assignments, HIT Lifetime (p. 26)
- Seeing HITs In Progress (p. 26)
- The Worker Submits, Returns or Abandons the Assignment (p. 26)
- Forcing a HIT to Expire Early (p. 27)
- Retrieving and Approving Results (p. 27)
- Reviewing HITs (p. 28)
- Paying the Worker a Bonus (p. 28)
- Disposing of the HIT (p. 29)
- Extending a HIT (p. 29)
- Disabling a HIT (p. 29)

A Worker Accepts a HIT

When your application creates a HIT using the CreateHITOperation operation, the HIT becomes available for Workers to find on the Amazon Mechanical Turk website. If a Worker has Qualifications that meet the HIT's Qualification requirements, the Worker can preview the HIT, then select the "Accept HIT" button to begin work.

When a Worker accepts a HIT, Amazon Mechanical Turk creates an *assignment* to track the work to completion. The assignment belongs exclusively to the Worker, and guarantees that the Worker will be allowed to submit results any time until the HIT's `AssignmentDurationInSeconds` has elapsed, and still be eligible for the reward.

By default, any Worker can accept a maximum of 10 assignments for your HITs. Once any Worker has accepted 10 of your assignments, they must complete one of the 10 assignments before Amazon Mechanical Turk lets them accept another one of your assignments.

Multiple Assignments, HIT Lifetime

By default, a HIT has at most one assignment. When a Worker accepts the HIT, an assignment is created, and the HIT is no longer available for other Workers to accept. If the Worker returns or abandons the HIT, the assignment is removed, and the HIT becomes available again.

A HIT can be created to accept multiple assignments by specifying a `MaxAssignments` parameter greater than 1 to the CreateHITOperation operation. Such a HIT will remain available for Workers to accept as long as the number of assignments, in progress or submitted, is less than `MaxAssignments`.

A HIT is only available for Workers to accept until the HIT's `LifetimeInSeconds` elapses, from the time the HIT was created. Once this time elapses, the HIT expires. Such a HIT is no longer available, even if the number of assignments is less than `MaxAssignments`. Workers with assignments in progress are allowed to continue working on the assignments as long as the `AssignmentDurationInSeconds` has not elapsed, even after the HIT expires. If a Worker returns or abandons the HIT after the HIT has expired, the HIT is not made available to other Workers.

Seeing HITs In Progress

You can retrieve a list of all of your HITs at any time using SearchHITs.

You can also retrieve the submitted assignments for a HIT at any time using GetAssignmentsForHIT. If your HIT has multiple assignments and has not expired, but some of the assignments have been submitted, GetAssignmentsForHIT will return the submitted assignments.

The Worker Submits, Returns or Abandons the Assignment

A Worker completes an assignment by entering values into the question form and selecting the "Submit HIT" button. The results are stored for later retrieval by your application. The Worker continues working on other HITs, or ends the session.

If a Worker decides not to complete a HIT after accepting it, the Worker may select the "Return HIT" button. Unless the HIT has expired-- that is, unless the HIT's `LifetimeInSeconds` has elapsed since the HIT was created-- the returned HIT becomes available for another qualified Worker to accept. The returned assignment ends, and is no longer accessible using the Requester API.

If a Worker does not submit results before the assignment's deadline-- that is, before the HIT's `AssignmentDurationInSeconds` has elapsed since the Worker accepted the HIT-- the HIT is considered abandoned. The effect is similar to if the Worker explicitly returned the HIT, except that abandonments and returns are tracked as separate statistics in the Worker's profile.

Tip

A Worker's HIT return and abandonment rates are tracked by Amazon Mechanical Turk as system Qualifications. You can create HITs that use Qualification requirements based on these numbers. Abandonments and returns are tracked as separate Qualifications.

Forcing a HIT to Expire Early

Normally, a HIT will remain available for Workers to accept as long as *MaxAssignments* results have not been submitted and the HIT has not expired (the HIT's *LifetimeInSeconds* has not elapsed).

You can cause a HIT to expire early by calling ForceExpireHIT. This has the same effect as the HIT expiring naturally: The HIT becomes no longer available for new Workers to accept. Assignments in progress are allowed to complete, either with the Worker submitting results, or the Worker returning or abandoning the assignment. After the HIT has expired, returned or abandoned assignments are not made available to other Workers.

Tip

ForceExpireHIT is your "stop" button. If you have submitted incorrect data for a HIT, or otherwise have changed your mind about the HIT, you can force it to expire to prevent Workers from completing it. You will still be responsible for approving assignments that have already been submitted, but ForceExpiredHIT can minimize the damage.

Whether the HIT expired with a call to ForceExpireHIT, or expired naturally, the HIT can be made available again with a call to ExtendHIT.

Retrieving and Approving Results

When either all of a HIT's assignments have been submitted by Workers, *or* the HIT has expired *and* all assignments have either been submitted, returned or abandoned, the HIT is considered "reviewable." Once a HIT is reviewable, the Requester can retrieve and process the results.

GetReviewableHITs returns the HIT IDs for all of your HITs that are reviewable. GetAssignmentsForHIT takes the ID of a reviewable HIT and returns the completed assignments, with the answers submitted by Workers. Your application can use these operations to poll for and retrieve results for HITs.

Once results for an assignment have been retrieved, you approve or reject the assignment. Your application calls ApproveAssignment to approve the assignment, or RejectAssignment to reject the assignment.

Note

A call to `ApproveAssignment` will return an error if the Requester's account does not have sufficient funds to pay the Worker and the listing fee at the time of the call.

An assignment should always be approved, unless it is clear the Worker did not follow the instructions in the HIT. The method your application uses to decide if an assignment should be approved will depend on the nature of your application. For example, you may request 3 assignments for each HIT then check to see if the results match, and if they do, approve all 3 assignments. If they don't match, have those results checked by a human operator. In some cases, it may be appropriate to automatically approve all assignments, then control the quality of answers using Qualification requirements.

Every HIT has an *AutoApprovalDelayInSeconds*, an amount of time after an assignment is submitted at which point Amazon Mechanical Turk will automatically approve the assignment if it has not already been explicitly approved or rejected. If not specified in the call to CreateHITOperation , this is set to the maximum, equivalent to 30 days. The maximum value assures that the Worker will get paid even if the Requester is unavailable. You may wish to specify a shorter auto-approval delay if the Workers are trusted (such as by having been vetted with Qualification requirements) and to assure Workers that they will be paid in a timely manner.

When you call ApproveAssignment, Amazon Mechanical Turk automatically processes payment of the reward to the Worker, and payment of the HIT listing fee, using the money in your Amazon.com account. You will not be charged for the reward, nor for the listing fee, if the results are rejected.

When you approve or reject an assignment, you can include a *RequesterFeedback* message, a string that the Worker can see on his or her HIT status screen. Including feedback message when rejecting a result may improve the quality of the results the Worker submits for your other HITs.

Reviewing HITs

When a HIT becomes reviewable, its assignments are ready for your application to retrieve and process. Your application can approve or reject assignments, then either dispose of the HIT, or extend it to collect more assignments. GetReviewableHITs returns the IDs of your HITs that have the "reviewable" status.

If your application does not want to make an immediate decision about approving or rejecting assignments, or disposing of or extending the HIT, the application can promote the status of the HIT to the "reviewing" status. "Reviewing" HITs are not returned by a call to GetReviewableHITs by default, so your application can continue to poll for "reviewable" HITs while "reviewing" HITs are awaiting processing. You can promote a HIT to the "reviewing" status using SetHITAsReviewing.

Tip

The "reviewing" status is useful for answer validation techniques that require waiting for additional information to approve an answer. For example, you can create a "verification" HIT based on the answers for a completed HIT that asks a Worker to compare the submitted answers and verify that they meet the HIT's requirements. Your application can give the original completed HIT a status of "reviewing" while it waits for the verification HIT to be completed, and continue to poll for other HITs as they become "reviewable".

Since your task HITs and your verification HITs will have different HIT types, your application can poll for task and verification results separately by passing the appropriate HIT type IDs to GetReviewableHITs.

You can retrieve a list of IDs of HITs with the "reviewing" status by calling GetReviewableHITs with an appropriate value for the *StatusMatchOption* parameter. You can revert a "reviewing" HIT back to the "reviewable" status by calling `SetHITAsReviewing` with an appropriate value for the *Revert* parameter.

Tip

There is no need to revert a "reviewing" HIT back to "reviewable" before disposing of or extending the HIT. If you have decided on the fate of the HIT, you can just call DisposedHIT or ExtendHIT on the "reviewing" HIT.

Paying the Worker a Bonus

When you approve an assignment a Worker has submitted, Amazon Mechanical Turk automatically processes the payment of the HIT's reward from your account to the Worker's account. By approving the assignment, you pay the Worker the amount of money specified in the original HIT.

In addition to the HIT reward, you can pay the Worker a "bonus" amount of money by calling GrantBonus. You might pay a bonus to reward Workers that submit high quality results. You could use the promise of a bonus payment to encourage Workers to complete your HITs and also to develop a following of Workers.

You can grant a bonus to any Worker who has submitted an assignment for one of your HITs any time after you approve or reject the assignment up until the HIT has been disposed.

Amazon Mechanical Turk collects a fee for bonus payments separate from (and in addition to) the HIT listing fee. The charge for awarding a bonus (currently) is 10% of the bonus with a minimum charge of $0.005. For example, if a Worker has a $0.01 assignment and is awarded a $0.01 bonus, the Requester pays the Worker $0.02 ($0.01 assignment+$0.01 bonus) and MTurk $0.01 ($0.005 + $0.005). Or, for example, if the Worker has a $1 assignment and is awarded a $1 bonus, the Requester pays the Worker $2 and MTurk $0.20.

For more information about Amazon Mechanical Turk pricing, go to the Amazon Mechanical Turk web site at Amazon Web Services.

Tip

There is no way to pay a Worker for completing a HIT an amount less than the reward for the HIT. When you approve the assignment, the Worker is paid the full HIT reward. Rejecting an assignment impacts your account statistics as well as the statistics of the Worker, so you should always approve an assignment if the Worker completed the instructions in the HIT successfully.

If you want to offer Workers a reward within a range of amounts, you can post the HIT with a reward equal to the lowest amount in the range, then mention the offer of a bonus payment in the description of the HIT. For example, to offer a HIT with a reward between $1 and $5, post the HIT with a reward of $1, then mention in the description that you will pay a bonus of up to $4.

You can retrieve a list of bonuses paid for a particular HIT or assignment by calling the `GetBonusPayments` operation.

Disposing of the HIT

Once all assignments have been approved or rejected, your application calls DisposeHIT. This removes the HIT from the list of HITs returned by GetReviewableHITs, and tells Amazon Mechanical Turk the data is no longer needed.

Extending a HIT

If a HIT has expired, or the maximum number of assignments have all been submitted, the HIT will no longer be available for Workers to accept. If the HIT has not gathered a satisfactory result, you may extend either the expiration date or the number of assignments, or both, using ExtendHIT. With appropriate values, extending the HIT will make it available again.

Disabling a HIT

ExtendHIT allows you to completely withdraw a HIT from the system, even before it has expired. Assignments that have been submitted (but not yet explicitly approved or rejected) will be approved automatically. Assignments in progress are allowed to complete, then approved automatically if submitted by the Worker. (Assignments returned or abandoned after the HIT is disabled are simply discarded.) The HIT and all submitted assignment data are disposed. A disabled HIT cannot be re-enabled.

Like ForceExpireHIT, disabling a HIT is useful if the HIT's question data is incorrect and would cause Workers to submit bad results. Because Workers expect to be paid for correctly answering the question,

even if the question is not the one you intended to ask, DisableHIT. This removes the HIT from the list of HITs returned by approves all assignments that have been submitted, and all assignments in progress that are submitted before their deadline. You will be charged for these assignments if you disable the HIT.

Tip

`DisableHIT` is like ForceExpireHIT, but much more drastic. It provides an automatic method to completely settle a HIT and all of its assignments with one service call.

Assignment data cannot be retrieved until the HIT enters the "reviewable" status, when it is no longer possible for Workers to submit more results. Because it is only necessary to disable a HIT to prevent Workers from submitting results, it is likely you will only wish to disable a HIT prior to it becoming "reviewable". This means you will not get to see the results for submitted assignments prior to disabling the HIT.

Creating and Managing Qualifications

Topics

- Qualifications and Qualification Requirements (p. 31)
- Qualification Requests (p. 32)
- Granting and Rejecting Qualification Requests (p. 33)
- Qualification Tests (p. 33)
- Assigning Qualifications Without Requests (p. 33)
- Updating and Retrieving Qualification Scores (p. 33)
- Revoking Qualifications (p. 34)
- Querying All Qualifications For a Type (p. 34)
- Querying All HITs That Use a Qualification Type (p. 34)
- Updating a Qualification Type's Test (p. 34)
- Deactivating a Qualification Type (p. 35)
- Disposing a Qualification Type (p. 35)
- System and Master Qualifications (p. 35)
- Searching Qualification Types (p. 35)

Qualifications and Qualification Requirements

A *Qualification* is a property of a Worker that represents a Worker's skill, ability or reputation. You can use Qualifications to control which Workers can perform your HITs. A HIT can have *Qualification requirements* that a Worker's Qualifications must meet before the Worker is allowed to accept the HIT. A requirement can also state that a Worker must meet the requirement to see the HIT's question data when previewing the HIT.

You can create and maintain your own Qualifications using the web service API. You create a *Qualification type* that describes what Qualifications of that type are for, and how Qualifications will be assigned.

A Qualification can have an optional value, such as a number. A HIT's Qualification requirement can specify a condition that the value must meet for the Qualification requirement to be met. A Worker can only accept a HIT if all of the HIT's Qualification requirements are met by the Worker's Qualifications.

You can use any Qualification type as the basis for your HIT's Qualification requirements, even if you did not create the type. For example, the Amazon Mechanical Turk system maintains a set of Qualification types that represent a Worker's account history, such as how many HITs the Worker has submitted in the lifetime of her account. You can create HIT requirements based on these statistics using the corresponding Qualification types.

You can create a new Qualification type by calling CreateQualificationType. Once a Qualification type has been created, it is available for use in a Qualification requirement, and can be searched or browsed at the Amazon Mechanical Turk web site.

Tip

Any Requester can use any Qualification type for a Qualification requirement, regardless of who created the type. However, only the creator of the Qualification type can grant requests for the type and assign values.

Qualification Requests

A Worker discovers a Qualification type by browsing HITs that require a Qualification of that type, or by searching or browsing the Qualification types directly. The Worker can view a description of the type, and request a Qualification of the type from the type's creator.

By default, a Worker can only request a Qualification for a given type once. The creator of the type can allow a Worker to re-request a Qualification by specifying a value for the `RetryDelayInSeconds` parameter when creating the type. If set, the Worker must wait until the delay has elapsed after the first request before requesting the Qualification again.

A Qualification type may include an optional *Qualification test*, a form similar to a HIT that the Worker must complete to request the Qualification. The data entered into this form can be collected by your application, used to determine whether or not the Qualification request should be granted, or used to determine the Qualification's score.

Qualification requests can be granted in three different ways:

- Your application can retrieve and process the requests for the Qualification. Your application downloads the Worker's answers for the Qualification test, and grants or rejects the request using the web service API.
- You can include an *answer key* with the Qualification test, which specifies desired answers and corresponding scores. Amazon Mechanical Turk processes Qualification requests for types with answer keys automatically. (Test answers are not available to your application in this case; the requests are granted immediately.)
- You can specify that the Qualification type should grant every request for the Qualification automatically, using a default score. You could use this to set up Qualifications your application will adjust later, based on the Worker's performance. (Auto-granted Qualifications do not have tests, and the requests are granted immediately.)

Tip

If you have a prior relationship with a Worker and know the Worker's ID, you can assign Qualifications directly to the Worker without the Worker first making a request. See the following discussion.

Granting and Rejecting Qualification Requests

Your application can retrieve pending requests for your Qualification types using GetQualificationRequests. To grant the request and (optionally) assign the Qualification's value, call GrantQualification. To reject the request, call RejectQualificationRequest.

Qualification Tests

A Qualification type may include a *Qualification test*, a set of questions similar to a HIT that a Worker must answer when requesting the Qualification. The test answers are returned with the request from a call to GetQualificationRequests. You can use the answers to determine the value to assign when granting the request.

The Qualification type may also include an *answer key* for the test, with answers and score values for each question in the test. In addition to score values for each matching answer, the answer key can specify how the Qualification value is calculated from the sum of the scores. For example, the Qualification value may be a percentage of correct answers.

Note

If a Qualification test has an answer key, the test may only contain multiple choice questions. An answer key cannot grade questions with free-text answers.

Tip

For more information on answer keys, see AnswerKey Data Structure.

Assigning Qualifications Without Requests

If you have a prior relationship with a Worker, such as if the Worker has submitted results for your HITs in the past, and you know the Worker's ID, you can assign a Qualification directly to the Worker without the Worker first making a request. AssignQualification gives a Worker a Qualification, and can send an e-mail message to the Worker saying that the Qualification was assigned. You can also tell Amazon Mechanical Turk to not send an e-mail message, and just assign the Qualification.

`AssignQualification` is useful for managing Qualifications for people that already do work for you on a regular basis. If you wish to establish new Qualifications for Workers that have never done work for you in the past, you can use HITs without Qualification requirements, or requestable Qualifications, to establish a work history.

Updating and Retrieving Qualification Scores

After you have granted a Qualification to a Worker and the Worker has completed a HIT for you, you can update the Worker's Qualification score with UpdateQualificationScoreO. The operation requires the Worker's user ID, which is included in the assignment data the Worker submits for your HITs. You can use this operation to revoke a qualifying score if the Worker is not performing to expectations, or promote a score if the Worker has earned it with good results.

You can query a user's Qualification score for a type you created using GetQualificationScore.

Note

If your Qualification type has a Qualification test, an answer key, and allows test retakes, a Worker who has been granted the Qualification can take the test again to get a new score, even if you have updated the score since it was first granted. To query a user's current score, use the `GetQualificationScore`. You can configure the ability to retake a Qualification test when you create the Qualification type.

Revoking Qualifications

You can revoke the Qualification at any time by calling RevokeQualification. A revoked Qualification behaves as if the Qualification were never granted: The Worker cannot qualify for HITs that require the Qualification be present. Also, calling the `GetQualificationScore` operation on a revoked Qualification will return an error.

A Worker may be able to request that a revoked Qualification be granted again if allowed by the Qualification type's retry policy. If the type does not have a retry delay, the Worker will not be able to request the Qualification a second time.

Querying All Qualifications For a Type

You can query all of the Qualifications you have granted for a type that you created using GetQualificationsForQualificationType. The operation returns the Worker ID and current Qualification score for every Qualification of the type that you have granted. You can request either all granted (active) Qualifications of the given type, or all revoked Qualifications of the given type.

Querying All HITs That Use a Qualification Type

You can query all of your HITs that use a given Qualification type in a Qualification requirement using GetHITsForQualificationType.

Note

You can only query your own HITs with `GetHITsForQualificationType`. The operation will not return HITs created by other Requesters that use a Qualification type.

You *can* query for HITs using any Qualification type, not only Qualification types that you created.

Updating a Qualification Type's Test

It is sometimes useful to make modifications to a Qualification type's test, to improve the accuracy of the resulting Qualification scores, or to replace old test questions with new ones. You can replace the test for a Qualification type using UpdateQualificationType.

`UpdateQualificationType` can also be used to replace an automatically graded test (with an answer key) with a manually graded one (without an answer key), and vice versa. It can also be used to change the test duration, the amount of time the Worker has to complete the test.

Deactivating a Qualification Type

Once you create a Qualification type, you are responsible for granting Qualification requests for the type (or having requests granted automatically using a Qualification test and answer key). You can temporarily deactivate a Qualification type by using `UpdateQualificationType` to update the status of the type. The type can be activated again with the same operation.

An inactive Qualification type cannot be requested by a Worker. It does not appear on the Amazon Mechanical Turk web site, and does not appear in search results. While the type is inactive, a HIT that requires a Qualification of the type cannot be created. Workers with Qualifications of the type will continue to have those Qualifications, and will continue to qualify for existing HITs that require the Qualification.

Disposing a Qualification Type

You can permanently remove a Qualification type by using DisposeQualificationType. Once a Qualification type has been disposed, you cannot use it to create HITs or HIT types. It does not appear on the web site and does not appear in search results. You can reuse the old Qualification type name and parameters to create a new Qualification type.

HITs that require a disposed Qualification type cannot be extended. If there are active HITs related to a disposed Qualification type, the HITs remain active and Workers who have qualified for that type can continue to work on those HITs. However, HIT types that use a disposed Qualification type can no longer be used to create new HITs.

Workers who have the Qualification of the disposed type keep those Qualifications, and will continue to be qualified for existing HITs that require the Qualification. If a Worker has a request pending to receive a Qualification for a disposed type, Amazon Amazon Mechanical Turk rejects the request.

System and Master Qualifications

Amazon Mechanical Turk provides a special set of Worker Qualifications available to all Requesters. System-assigned qualifications include approval rate and location. The system assigns these Qualifications to every Worker and updates their values as Workers use the system. For a list of system Qualifications and more information about locale-based Qualification requirements, go to the QualificationRequirement Data Structure, in the *Amazon Mechanical Turk API Reference*.

Mechanical Turk Masters

Mechanical Turk also provides Requesters access to Mechanical Turk Masters. Masters have demonstrated accuracy in completing HITs in the Mechanical Turk marketplace. There are two types of Master Qualifications: Photo Moderation Masters and Categorization Masters.

Searching Qualification Types

SearchQualificationTypes returns Qualification types whose names or descriptions match a search query. The results are similar to what is returned when performing such a search on the Amazon Mechanical Turk web site or Requester Console, except that system Qualifications may also be included in results.

Creating and Managing Notifications

Tracking Your HITs

After your application creates a HIT, Amazon Mechanical Turk manages the events that lead to the HIT's completion. In many cases, your application only needs to call GetReviewableHITs periodically until the HIT is returned as a result. The HIT becomes "reviewable" either when all of the HIT's assignments have been completed by Workers, or the HIT's lifetime has elapsed (the HIT has expired).

Sometimes it is useful to know more about the events that lead to the HIT becoming reviewable. One way to watch a HIT and its assignments change state is to call GetHIT periodically, checking the values of the `HITStatus` and assignment summary fields. Depending on how up-to-date you need the information to be, doing this would require downloading the HIT's field data many times, and most attempts would only tell you that nothing has happened since the last call.

The Amazon Mechanical Turk Requester Service provides a better way to keep track of HIT activity. You can set up *notifications* for any HIT type, and any of several kinds of events that occur during the HIT life cycle. When Amazon Mechanical Turk detects an event for which you've set up a notification, Amazon Mechanical Turk will attempt to notify you--or your application--that the event took place.

Setting Notifications

Notifications are specified as part of a HIT type. You can create or modify notifications for a HIT type with SetHITTypeNotification. The operation accepts a notification specification, as the `Notification` parameter.

The HIT type must already exist before you can give it a notification specification. You can create a new HIT type using RegisterHITType You can determine the HIT type of an existing HIT by calling GetHIT with the HIT's ID.

You can update, disable or enable a HIT type's notification specification at any time with SetHITTypeNotification.

Ways To Be Notified

The simplest way to be notified of an event is by e-mail. If you establish a notification for an event using e-mail as the transport, when the event occurs, Amazon Mechanical Turk will send a human-readable e-mail message to an address you include in the notification specification.

Amazon Mechanical Turk can also send notifications directly to your application using a web service call. To accept notifications, your application must be connected to the Internet, and must be able to accept HTTP requests. Amazon Mechanical Turk can send a notification as either a SOAP web service message, or a REST-like HTTP request. For more information about developing a HIT notification receptor, see Notification Receptor API.

Notifications and Events

Amazon Mechanical Turk can notify you when:

- A Worker accepts a HIT
- A Worker abandons an assignment
- A Worker returns an assignment
- A Worker submits an assignment
- A HIT becomes "reviewable"
- A HIT expires

For example, instead of calling GetReviewableHITs repeatedly to poll for results of a particular HIT type, you can establish a notification that triggers whenever a HIT of that type becomes "reviewable". Your application can listen for a SOAP or REST request that indicates when a HIT of the type is ready for review.

Notifications and Reliability

A notification message is only sent once. If your application is not available when the notification is sent, the notification is not sent a second time.

A notification message for an event is sent within several minutes of the event occurring. If many events occur in a short period of time, Amazon Mechanical Turk will send a single notification message that describes multiple events.

You can test your application's ability to receive notifications using SendTestEventNotification. When you call this operation, Amazon Mechanical Turk will send a test notification according to the notification specification you provide.

Notifications and Security

For a notification sent as a SOAP or REST message, the message includes a deprecated signature element that you should not make assertions about. For secure notifications, please use the Amazon Simple Queue Service (Amazon SQS) in the latest version of the WSDL.

For more information about developing a HIT notification receptor, see Notification Receptor API.

Document History

This Document History describes the important changes to the documentation in this release of Amazon Mechanical Turk.

Relevant dates to this History

- **Current product version**-2011-10-01
- **Latest product release**-December 2011
- **Latest documentation update**-01 December 2011

Change	Description	Release Date
New Mechanical Turk Review Policies	Amazon Mechanical Turk has added Review Policies that you can use to evaluate Worker submissions against a defined set of criteria. For more information, see Review Policies.	In this Release
Mechanical Turk Masters Qualifications	Amazon Mechanical Turk has added two new Mechanical Turk Masters Qualification types: Categorization Masters and Photo Moderation Masters. The new Mechanical Turk Masters role is established for an elite group of Workers who have demonstrated accuracy on specific types of HITs on the Mechanical Turk marketplace. For more information, see Creating and Managing Qualifications (p. 31).	2011-06-22
Technical documents reorganized	The API reference and the command line tool reference have been split out of the Amazon Mechanical Turk Developer Guide. Now, on the documentation landing page, Amazon Mechanical Turk Documentation you can select the document you want to view. When viewing the documents online, the links in one document will take you, when appropriate, to one of the other guides.	2009-09-16