Amazon Mechanical Turk

Legacy API Reference

API Version 2014-08-15

Amazon Mechanical Turk: Legacy API Reference

Table of Contents

Amazon Mechanical Turk Legacy API Reference

This is the *Amazon Mechanical Turk Legacy API Reference*. This guide provides detailed information about old Amazon Mechanical Turk operations, data structures, and parameters. The major sections of this guide are described in the following table.

Check out the newest API reference *Amazon Mechanical Turk Legacy API Reference* (p. 1)

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 the Amazon Mechanical Turk website.

Operations (p. 10)	Alphabetical list of all Amazon Mechanical Turk operations.
Data Structures (p. 140)	Alphabetical list of all Amazon Mechanical Turk data structures.
Common Parameters (p. 6)	Descriptions of the parameters common to all operations.
WSDL and Schema Locations (p. 2)	Links to Amazon Mechanical Turk WSDL and schemas.
Review Policies (p. 178)	Description of Amazon Mechanical Turk Review Policies.
Question and Answer Data (p. 190)	Description of question and answer data that Amazon Mechanical Turk passes between Requesters and Workers.
The Notification API (p. 225)	Description of how Amazon Mechanical Turk sends notification messages to your application.

WSDL and Schema Locations

Topics

The Amazon Mechanical Turk Service can be accessed using the SOAP web services messaging protocol, or using the REST method of HTTP requests with parameters. The SOAP interface is described by a Web Services Description Language (WSDL) document. REST requests return messages that conform to an XML Schema document.

To make it easy to upgrade your application when a new version of schemas are released, all schemas have a version number. The version number appears in the URL of a schema file, and in a schema's target namespace. The API schemas (the WSDL and request/response messages) and the data structure schemas (question and answer values) use separate version numbers. The latest versions are as follows:

Type of Schema	Latest Version
The API: WSDL and message schemas	**2014-08-15**
The HTMLQuestion schema	**2011-11-11**
The QuestionForm, QuestionFormAnswers and AnswerKey schemas	**2005-10-01**
The ExternalQuestion schema	**2006-07-14**
The formatted content XHTML subset	**2006-07-14**
The Notification API	**2006-05-05**

API Release Notes

See the API release notes for new features and bug fixes.

The WSDL and Message Schema Locations

The WSDL for a given version of the Amazon Mechanical Turk Service API can be found at a URL that corresponds to the API version. For example, the WSDL for the **2014-08-15** version of the API can be found here:

http://mechanicalturk.amazonaws.com/AWSMechanicalTurk/2014-08-15/AWSMechanicalTurkRequester.wsdl

The XML Schema for the messages of a given version of the Amazon Mechanical Turk Service API can be found at a URL that corresponds to the API version. For example, the XML Schema for the **2014-08-15** version of the API can be found here:

http://mechanicalturk.amazonaws.com/AWSMechanicalTurk/2014-08-15/AWSMechanicalTurkRequester.xsd

The Data Structure Schema Locations

The Amazon Mechanical Turk Service has several parameters and return values that contain XML data. The XML content must validate against the appropriate XML schema. For more information, see HTMLQuestion (p. 220), QuestionForm (p. 191), QuestionFormAnswers (p. 211), and AnswerKey (p. 212).

Note
The API version number and the data structure version number are not related. The two sets of schemas may have new releases at different times, and may have different version numbers. For example, an application using the **2014-08-15** version of the API may create HITs using the **2005-10-01** version of the QuestionForm schema. (There may not be a "2014-08-15" version of the QuestionForm schema.)
Your application may use any supported version of the data schemas with any supported version of the API. A data structure returned by the service will include a namespace that corresponds to the relevant schema.

The **2011-11-11** version of the HTMLQuestion schema can be found here:

http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2011-11-11/HTMLQuestion.xsd

The **2005-10-01** version of the QuestionForm schema can be found here:

http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2005-10-01/QuestionForm.xsd

The **2005-10-01** version of the QuestionFormAnswers schema can be found here:

http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2005-10-01/QuestionFormAnswers.xsd

The **2005-10-01** version of the AnswerKey schema can be found here:

http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2005-10-01/AnswerKey.xsd

The **2006-07-14** version of the ExternalQuestion schema can be found here:

http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2006-07-14/ExternalQuestion.xsd

Note
To conform to a schema, XML content must use namespace declarations that match the target namespace for the schema. The target namespace is declared at the top of the schema, as the "targetNamespace" attribute of the "xs:schema" element.
The schemas for QuestionForm, QuestionFormAnswers, and AnswerKey use namespace URIs similar to the URL at which the schema file can be found, including the service version. For example:

```
<QuestionForm
  xmlns="http://mechanicalturk.amazonaws.com/
AWSMechanicalTurkDataSchemas/2005-10-01/QuestionForm.xsd"
```

```
  >
    [...]
</QuestionForm>
```

If the service returns an error message about data not validating against the schema, make sure your namespace declaration matches the target namespace specified in the schema.

The Formatted Content XHTML Subset

HITs and Qualification tests can include blocks of content formatted with XHTML tags in their instructions and question data. To include text and markup for formatted content in a web service request, you specify it as XML CDATA inside a `FormattedContent` element, part of the QuestionForm data structure (p. 191).

Amazon Mechanical Turk validates formatted content by converting the text and markup in the CDATA block into an XML document, then validating it against a schema. For more information about how this XML document is produced, see Formatted Content: XHTML (p. 206), "How XHTML Formatted Content Is Validated".

The **`2006-07-14`** version of the schema used to validate formatted content can be found here:

http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2006-07-14/FormattedContentXHTMLSubset.xsd

The Notification API WSDL Location

The WSDL for the Notification API is located at:

http://mechanicalturk.amazonaws.com/AWSMechanicalTurk/2006-05-05/AWSMechanicalTurkRequesterNotification.wsdl

For more information about the Notification API, see The Notification API (p. 225).

Service API Versions

When a new version of the service API is released, previous versions are supported for a limited time to allow applications to continue to function until they are upgraded. The version of a service API is specified as a date, such as **`2014-08-15`**.

The version of the API can be found in the URLs of the WSDL and schema files. It can also be found in the `targetNamespace` of the WSDL and schema files.

You can retrieve the WSDL or schema files for previous versions of the API by replacing the version date in the URL with the desired version. For example, to retrieve the WSDL for API version **`2014-08-15`**:

http://mechanicalturk.amazonaws.com/AWSMechanicalTurk/2014-08-15/AWSMechanicalTurkRequester.wsdl

Accessing a Specific Service Version

For your application to use a specific version of the service API, the service needs to be told which version is being used with each request.

For SOAP requests, the Amazon Mechanical Turk Service determines which API version you are using based on the namespace in your request message, which is determined by the WSDL you are using with your application. SOAP requests always include this information, and SOAP toolkits determine the namespace automatically from the WSDL.

For REST requests, you must explicitly request the version to use by including the `Version` parameter in your request. The `Version` parameter ensures that the service does not return response elements that your application is not designed to handle.

Here is an example REST request that includes the `Version` parameter:

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2014-08-15
&Operation=GetHIT
&HITId=123RVWYBAZW00EXAMPLE
```

The Default Version

Older AWS services supported requests that did not specify an API version. This behavior is still supported for legacy reasons, but its use is discouraged.

When the Amazon Mechanical Turk Service receives a REST request without a `Version` parameter, the service will use the latest version. If your application does not specify the `Version` in each request, when a new version of the API is released, your application will start using the new version automatically. Because new versions of the API may be incompatible with applications expecting to use an older version, specifying an explicit `Version` parameter with each request is strongly recommended.

A similar legacy feature exists for SOAP: A request for the WSDL or a schema file using a URL that does not include the version number will return the file for the latest version of the API. Using WSDL/schema URLs that include the API version number is strongly recommended.

Common Parameters

Topics

The Amazon Mechanical Turk Service accepts a set of parameters in the request common to every operation. Each required parameter must be included in a request for the request to be successful. Parameters common to all operations are explained in **Common Request Parameters**. For more information about the parameters for a specific operation, see the description of the operation in the Operations (p. 10) section of this reference.

Response groups specify what data is returned by Mechanical Turk for an operation request and are explained in Response Groups (p. 7).

Common Request Parameters

Requests to the Amazon Mechanical Turk service can include the parameters described in the following table. Required parameters must be included with each request for the request to succeed.

Name	Description	Required
`AWSAccessKeyId`	The Requester's Access Key ID, a unique identifier that corresponds to a Secret Access Key and an Amazon.com account. Type: String Default: None	Yes
`Service`	The name of the Amazon Web Services service. Type: String Valid Values: AWSMechanicalTurkRequester Default: None Constraints: For REST requests only. For SOAP requests the name of the service is part of the SOAP entry point, and does not need to be specified in the request.	Yes
`Operation`	The name of the operation. Type: String Default: None Contraints: For REST requests only	Yes

Name	Description	Required
	For SOAP requests, the operation name is part of the SOAP message structure provided by your SOAP toolkit, and is not part of the request.	
`Signature`	The signature for this request, an encrypted string calculated from elements of the request and the AWS access key that corresponds to your AWS Access Key ID. For information about how to calculate a `Signature`, see AWS Request Authentication. Type: String Default: None	Yes
`Timestamp`	The current time on your system. This value is included to validate against the `Signature` parameter. Type: a dateTime in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **`2005-01-31T23:59:59Z`**. Default: None	Yes
`ResponseGroup`	A list of response groups. For more information about response groups, see Response Groups (p. 7). Type: String Default: None	Yes
`Version`	Specifies what version of the API to use. Type: String Default: None. If not specified, the latest version of the API is used. Constraints: Used only for REST requests	No
`Validate`	*`Deprecated`*	No
`Credential`	This parameter is reserved for future use. Type: None Default: None	Not used

Response Groups

Response groups specify what data is returned by the service for an operation, to control the data included in the response. Most operations in the Amazon Mechanical Turk service allow the use of two common response groups: `Minimal` and `Request`. `Minimal` provides a minimal set of results of the operation call. `Request` echoes the content of the original request. What gets included in `Minimal` varies depending upon the API call.

The following table lists the operations that can return more response groups than `Minimal` and `Request`.

Operation	Allowable Response Groups	Default Response Groups
CreateHit	Request, Minimal, HITDetail, HITQuestion, HITAssignmentSummary	Minimal
GetAssignment	Request, Minimal, AssignmentFeedback, HITDetail, HITQuestion	Minimal
GetAssignmentsForHIT	Request, Minimal, AssignmentFeedback	Minimal
GetHIT	Request, Minimal, HITDetail, HITQuestion, HITAssignmentSummary	Minimal, HITDetail, HITQuestion
GetRequesterStatistic	Request, Minimal, Parameters	Minimal, Parameters
GetRequesterWorkerStatistic	Request, Minimal, Parameters	Minimal, Parameters
SearchHITS	Request, Minimal, HITDetail, HITQuestion, HITAssignmentSummary	Minimal, HITDetail, HITAssignmentSummary

Response Groups Content

The following table lists the elements returned in the response groups other than the Minimal and Request groups. The table also lists the response group elements associated with each response group.

Response Group	Response Group Elements (alpha order)
AssignmentFeedback	RequesterFeedback For more information on this element, see the Assignment (p. 140) data structure.
HITAssignmentSummary	NumberOfAssignmentsAvailable NumberOfAssignmentsCompleted NumberOfAssignmentsPending For more information on these elements, see the HIT (p. 144) data structure.
HITDetail	AssignmentDurationInSeconds AutoApprovalDelayInSeconds CreationTime Description Expiration Keywords HITGroupId HITLayoutId

Response Group	Response Group Elements (alpha order)

	`HITReviewStatus`
	
	`HITStatus`
	
	`MaxAssignments`
	
	`QualificationRequirement`
	
	`RequesterAnnotation`
	
	`Reward`
	
	`Title`
	
	For more information on these elements, see the HIT (p. 144) data structure.

`HITQuestion`	`Question` For more information on this elements, see the HIT (p. 144) data structure.

`Parameters`	`Statistic`
	
	`TimePeriod`

Operations

The Amazon Mechanical Turk API consists of web service operations for every task the service can perform. This section describes each operation in detail.

ApproveAssignment

Description

The `ApproveAssignment` operation approves the results of a completed assignment.

Approving an assignment initiates two payments from the Requester's Amazon.com account: the Worker who submitted the results is paid the reward specified in the HIT, and Amazon Mechanical Turk fees are debited. If the Requester's account does not have adequate funds for these payments, the call to `ApproveAssignment` returns an exception, and the approval is not processed.

You can include an optional feedback message with the approval, which the Worker can see in the **Status** section of the web site.

Request Parameters

The `ApproveAssignment` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `ApproveAssignment` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: ApproveAssignment Default: None	Yes
`AssignmentId`	The ID of the assignment. This parameter must correspond to a HIT created by the Requester. Type: String Default: None	Yes
`RequesterFeedback`	A message for the Worker, which the Worker can see in the **Status** section of the web site. Type: String Default: None Constraints: Can be up to 1024 characters (including multi-byte characters). The `RequesterFeedback` parameter cannot contain ASCII characters 0-8, 11,12, or 14-31. If these characters are present, the operation throws an InvalidParameterValue error.	No

Response Elements

A successful request for the `ApproveAssignment` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`ApproveAssignmentResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `ApproveAssignment` operation.

Sample Request

The following example approves an assignment identified by its assignment ID.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=ApproveAssignment
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&AssignmentId=123RVWYBAZW00EXAMPLE456RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<ApproveAssignmentResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</ApproveAssignmentResult>
```

ApproveRejectedAssignment

Description

The `ApproveRejectedAssignment` operation approves an assignment that was previously rejected.

`ApproveRejectedAssignment` works only on rejected assignments that were submitted within the previous 30 days and only if the assignment's related HIT has not been disposed.

Approving the rejected assignment initiates two payments from the Requester's Amazon.com account: one payment to the Worker who submitted the results for the reward amount specified in the HIT and one payment for Amazon Mechanical Turk fees. For the operation to succeed, a Requester must have sufficient funds in their account to pay the Worker and the fees.

If the assignment is not currently rejected, or if the Requester does not have sufficient funds in their account to pay the Worker and the Mechanical Turk fees, then the `ApproveRejectedAssignment` operation returns an exception and the approval is not processed.

You can include an optional feedback message with the approval, which the Worker can see in the Status section of the Amazon Mechanical Turk website.

Request Parameters

The `ApproveRejectedAssignment` operation accepts parameters common to all operations. Some common parameters are required. For more information, see Common Parameters (p. 6).

The following parameters are specific to the `ApproveRejectedAssignment` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: ApproveRejectedAssignment Default: None	Yes
`AssignmentId`	The ID of the assignment. This parameter must correspond to a HIT created by the Requester. Type: String Default: None.	Yes
`RequesterFeedback`	A message for the Worker, which the Worker can see in the Status section of the Mechanical Turk website. Type: String Default: None. Constraints: Can be up to 1024 characters (including multi-byte characters). The `RequesterFeedback` parameter cannot contain ASCII characters 0-8, 11,12, or 14-31. If these	No

Name	Description	Required
	characters are present, the operation throws an **InvalidParameterValue** error.	

Response Elements

A successful request for the `ApproveRejectedAssignment` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`ApproveRejectedAssignmentResult`	Contains a Request element if the **Request** `ResponseGroup` is specified.

Examples

The following example shows how to use the `ApproveRejectedAssignment` operation.

Sample Request

The following example approves a previously rejected assignment identified by its assignment ID.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=ApproveRejectedAssignment
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&AssignmentId=123RVWYBAZW00EXAMPLE456RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<ApproveRejectedAssignmentResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</ApproveRejectedAssignmentResult>
```

AssignQualification

Description

The `AssignQualification` operation gives a Worker a Qualification. `AssignQualification` does not require that the Worker submit a Qualification request. It gives the Qualification directly to the Worker.

You can only assign a Qualification of a Qualification type that you created (using the CreateQualificationType (p. 32) operation).

Tip
`AssignQualification` does not affect any pending Qualification requests for the Qualification by the Worker. If you assign a Qualification to a Worker, then later grant a Qualification request made by the Worker, the granting of the request may modify the Qualification score. To resolve a pending Qualification request without affecting the Qualification the Worker already has, reject the request with the RejectQualificationRequest (p. 113) operation.

Request Parameters

The `AssignQualification` operation accepts parameters common to all operations. Some common parameters are required. See CommonParameters (p. 6) for more information.

The following parameters are specific to the `AssignQualification` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: AssignQualifcation Default: None	Yes
`QualificationTypeId`	The ID of the Qualification type to use for the assigned Qualification. Type: String Default: None Constraints: must be a valid Qualification type ID, as returned by the CreateQualificationType (p. 32) operation.	Yes
`WorkerId`	The ID of the Worker to whom the Qualification is being assigned. Worker IDs are included with submitted HIT assignments and Qualification requests. Type: String Default: None	Yes
`IntegerValue`	The value of the Qualification to assign. Type: Integer Default: 1	No
`SendNotification`	Specifies whether to send a notification email message to the Worker saying that the qualification was assigned to the Worker. Type: Boolean Valid Values: `true` | `false`. Default: `true`	No

Response Elements

A successful request for the AssignQualification operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`AssignQualificationResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `AssignQualification` operation.

Sample Request

The following example assigns a Qualification of a specified type to a Worker with the specified ID, using the specified Qualification value. By default, Amazon Mechanical Turk sends the Worker an e-mail message saying that the Worker has received the Qualification.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=AssignQualification
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&QualificationTypeId=789RVWYBAZW00EXAMPLE
&WorkerId=AZ3456EXAMPLE
&IntegerValue=800
```

Sample Response

The following is an example response.

```
<AssignQualificationResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</AssignQualificationResult>
```

BlockWorker

Description

The `BlockWorker` operation allows you to prevent a Worker from working on your HITs. For example, you can block a Worker who is producing poor quality work. You can block up to 100,000 Workers.

> **Note**
> `BlockWorker` prevents a Worker from accepting more of your HITs after you block them. However, `BlockWorker` does not prevent a Worker from submitting assignments that they accepted before you blocked them.

You need the Worker ID to use this operation. You can get the Worker ID in the assignment data returned by a call to the GetAssignmentsForHIT (p. 52) operation. If the Worker ID is missing or invalid, this operation returns with the failure message "WorkerId is invalid." If the Worker is already blocked, this operation returns successfully.

Request Parameters

The `BlockWorker` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `BlockWorker` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: BlockWorker Default: None	Yes
`WorkerId`	The ID of the Worker to block. Type: String Default: None	Yes
`Reason`	A message explaining the reason for blocking the Worker. This parameter enables you to keep track of your Workers. The Worker does not see this message. Type: String Default: None	Yes

Response Elements

A successful request for the `BlockWorker` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`BlockWorkerResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `BlockWorker` operation.

Sample Request

The following example blocks a Worker from working on your HITs.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=BlockWorker
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&WorkerId=AZ3456EXAMPLE
&Reason=After%20several%20warnings,%20he%20continued%20to%20submit%20answers%20without
%20reading%20the%20instructions%20carefully.
```

Sample Response

The following is an example response.

```
<BlockWorkerResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</BlockWorkerResult>
```

Related Operations

To unblock a Worker use the UnblockWorker (p. 132) operation.

ChangeHITTypeOfHIT

Description

The `ChangeHITTypeOfHIT` operation allows you to change the `HITType` properties of a HIT. This operation disassociates the HIT from its old `HITType` properties and associates it with the new `HITType` properties. The HIT takes on the properties of the new `HITType` in place of the old ones. For more information about HIT types, see the Amazon Mechanical Turk Developer Guide.

You can use `ChangeHITTypeOfHIT` to update any of the `HITType` properties of a HIT.

Request Parameters

The `ChangeHITTypeOfHIT` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `ChangeHITTypeOfHIT` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: ChangeHITTypeOfHIT Default: None	Yes
`HITId`	The ID of the HIT to change Type: String Default: None	Yes
`HITTypeId`	The ID of the new HIT type Type: String Default: None	Yes

Response Elements

A successful request for the `ChangeHITTypeOfHIT` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`ChangeHITTypeOfHITResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `ChangeHITTypeOfHIT` operation.

Sample Request

The following example changes the HIT type.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=ChangeHITTypeOfHIT
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITId=123RVWYBAZW00EXAMPLE
&HITTypeId=T100CN9P324W00EXAMPLE
```

Sample Response

The following is an example response.

```
<ChangeHITTypeOfHITResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</ChangeHITTypeOfHITResult>
```

CreateHIT

Description

The `CreateHIT` operation creates a new Human Intelligence Task (HIT). The new HIT is made available for Workers to find and accept on the Amazon Mechanical Turk website.

There are two ways to specify HIT properties when calling the `CreateHIT` operation: with the HIT type ID, or with the common property values. If the `HITTypeId` parameter is specified, the `CreateHIT` operation assumes the syntax with a HIT type ID is what you intended. If you provide both a HIT type ID and values for the common properties, the common property values are ignored.

CreateHIT also supports several ways to provide question data: by providing a value for the `Question` parameter that fully specifies the contents of the HIT, or by providing a HitLayoutId and associated HitLayoutParameters. If you are providing a data structure, it may be a QuestionForm (p. 191) structure, an ExternalQuestion (p. 216) structure, or an HTMLQuestion (p. 220) structure. For more information, see the `Question` parameter.

Using Review Policies with `CreateHIT` you can specify how you want Mechanical Turk to take action on the HITs you create. Using Review Policies removes the need for you to manually take action and determine Worker accuracy after the HIT has been completed. For more information about Review Policies, see Review Policies (p. 178).

You must specify review policies when you create a HIT. You cannot apply a Review Policy to an existing HIT. There are two types of Review Policies, Assignment-level and HIT-level:

- An Assignment-level Review Policy is applied as soon as a Worker submits an assignment. For more information, see Assignment Review Policies (p. 179).
- A HIT-level Review Policy is applied when a HIT becomes reviewable. For more information, see HIT Review Policies (p. 181).

You can specify one Assignment-level Review Policy and one HIT-level Review Policy when you call `CreateHIT` using the HIT Review Policy (p. 151) data structure. The Assignment-level Review Policy ScoreMyKnownAnswer/2011-09-01 and the HIT-level Review Policy SimplePlurality/2011-09-01 can be used in the same call to `CreateHIT`.

- Use the ScoreMyKnownAnswers/2011-09-01 policy parameters to compare a known answer you provide with the answer a Worker provides. Mechanical Turk will automatically compare the answer and take action based on your requirements.
- Use the SimplePlurality/2011-09-01 policy parameters to automatically compare answers received from multiple workers and detect if there is a majority or consensus answer.

For more information about how to get the results you need from Workers, see Review Policy Use Cases (p. 185).

> **Note**
> If a HIT is created with 10 or more maximum assignments, there is an additional fee. For more information, see Amazon Mechanical Turk Pricing.

Request Parameters

The `CreateHIT` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following tables describe the additional parameters required to call the `CreateHIT` operation with an explicit HIT type ID and without a HIT type ID.

Calling CreateHIT with a HIT Type ID

The following parameters are specific to calling the `CreateHIT` operation with an explicit HIT type ID:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: CreateHIT Default: None	Yes
`HITTypeId`	The HIT type ID. Type: String Default: None	Yes
`Question`	The data the person completing the HIT uses to produce the results. Type: String Default: None Constraints: Must be a QuestionForm (p. 191) data structure, an ExternalQuestion (p. 216) data structure, or an HTMLQuestion (p. 220) data structure. The XML question data must not be larger than 64 kilobytes (65,535 bytes) in size, including whitespace. Either a `Question` parameter or a `HITLayoutId` parameter must be provided.	No
`HITLayoutId`	The `HITLayoutId` allows you to use a pre-existing HIT design with placeholder values and create an additional HIT by providing those values as `HITLayoutParameters`. For more information, see HITLayout (p. 223). Type: String Default: None Constraints: Must be a valid `HITLayoutId`, as obtained from the Amazon Mechanical Turk Requester website. Either a `Question` parameter or a `HITLayoutId` parameter must be provided.	No
`HITLayoutParameter`	If the `HITLayoutId` is provided, any placeholder values must be filled in with values using the `HITLayoutParameter` structure. For more information, see HITLayout (p. 223). Type: HITLayoutParameter (p. 150)	No

Name	Description	Required
	Default: None	
`LifetimeInSeconds`	The number of seconds after which the HIT is no longer available for users to accept. After the lifetime of the HIT has elapsed, the HIT no longer appears in HIT searches, even if not all of the HIT's assignments have been accepted. Type: positive integer Valid Values: any integer between 30 (30 seconds) and 31536000 (365 days). Default: None	Yes
`MaxAssignments`	The number of times the HIT can be accepted and completed before the HIT becomes unavailable. Type: positive integer Valid Values: any integer between 1 and 1000000000 (1 billion). Default: 1	No
`AssignmentReviewPolicy`	The Assignment-level Review Policy applies to the assignments under the HIT. You can specify for Mechanical Turk to take various actions based on the policy. For more information, see Assignment Review Policies (p. 179). Type: String Valid Values: Must be a HIT Review Policy data structure, see HIT Review Policy (p. 151). Default: none	No
`HITReviewPolicy`	The HIT-level Review Policy applies to the HIT. You can specify for Mechanical Turk to take various actions based on the policy. For more information, see HIT Review Policies (p. 181). Type: String Valid Values: Must be a HIT Review Policy data structure, see HIT Review Policy (p. 151). Default: none	No

Name	Description	Required
`RequesterAnnotation`	An arbitrary data field. The `RequesterAnnotation` parameter lets your application attach arbitrary data to the HIT for tracking purposes. For example, this parameter could be an identifier internal to the Requester's application that corresponds with the HIT. The `RequesterAnnotation` parameter for a HIT is only visible to the Requester who created the HIT. It is not shown to the Worker, or any other Requester. The `RequesterAnnotation` parameter may be different for each HIT you submit. It does not affect how your HITs are grouped. Type: String Default: None Constraints: must not be longer than 255 characters in length.	No
`UniqueRequestToken`	A unique identifier for this request. Allows you to retry the call on error without creating duplicate HITs. This is useful in cases such as network timeouts where it is unclear whether or not the call succeeded on the server. If the HIT already exists in the system from a previous call using the same `UniqueRequestToken`, subsequent calls will return a *AWS.MechanicalTurk.HitAlreadyExists* error with a message containing the HITId. Type: String Default: None Constraints: must not be longer than 64 characters in length. Note: It is your responsibility to ensure uniqueness of the token. The unique token expires after 24 hours. Subsequent calls using the same `UniqueRequestToken` made after the 24 hour limit could create duplicate HITs.	No

Calling CreateHIT Without a HIT Type ID

The following parameters are specific to calling the `CreateHIT` operation without a HIT type ID, letting Amazon Mechanical Turk determine the HIT type from the property values:

Name	Description	Required
`Operation`	The name of the operation Type: String	Yes

Name	Description	Required
	Valid Values: CreateHIT Default: None	
`Title`	The title of the HIT. A title should be short and descriptive about the kind of task the HIT contains. On the Amazon Mechanical Turk web site, the HIT title appears in search results, and everywhere the HIT is mentioned. Type: String Default: None Constraints: must not be more than 128 characters	Yes
`Description`	A general description of the HIT. A description includes detailed information about the kind of task the HIT contains. On the Amazon Mechanical Turk web site, the HIT description appears in the expanded view of search results, and in the HIT and assignment screens. A good description gives the user enough information to evaluate the HIT before accepting it. Type: String Default: None Constraints: cannot be more than 2,000 characters in length	Yes
`Question`	The data the person completing the HIT uses to produce the results. Type: String Default: None Constraints: Must be a QuestionForm (p. 191) data structure, an ExternalQuestion (p. 216) data structure, or an HTMLQuestion (p. 220) data structure. The XML question data must not be larger than 64 kilobytes (65,535 bytes) in size, including whitespace. Either a `Question` parameter or a `HITLayoutId` parameter must be provided.	No

Name	Description	Required
HITLayoutId	The HITLayoutId allows you to use a pre-existing HIT design with placeholder values and create an additional HIT by providing those values as HITLayoutParameters. For more information, see HITLayout (p. 223). Type: String Default: None Constraints: Must be a valid HITLayoutId, as obtained from the Amazon Mechanical Turk Requester website. Either a Question parameter or a HITLayoutId parameter must be provided.	No
HITLayoutParameter	If the HITLayoutId is provided, any placeholder values must be filled in with values using the HITLayoutParameter structure. For more information, see HITLayout (p. 223). Type: HITLayoutParameter (p. 150) Default: None	No
Reward	The amount of money the Requester will pay a Worker for successfully completing the HIT. Type: Price (p. 157) data structure. Default: None	Yes
AssignmentDurationInSeconds	The amount of time, in seconds, that a Worker has to complete the HIT after accepting it. If a Worker does not complete the assignment within the specified duration, the assignment is considered abandoned. If the HIT is still active (that is, its lifetime has not elapsed), the assignment becomes available for other users to find and accept. Type: positive integer Valid Values: any integer between 30 (30 seconds) and 31536000 (365 days). Default: None	Yes

Name	Description	Required
`LifetimeInSeconds`	An amount of time, in seconds, after which the HIT is no longer available for users to accept. After the lifetime of the HIT elapses, the HIT no longer appears in HIT searches, even if not all of the assignments for the HIT have been accepted. Type: positive integer Valid Values: any integer between 30 (30 seconds) and 31536000 (365 days). Default: None	Yes
`Keywords`	One or more words or phrases that describe the HIT, separated by commas. These words are used in searches to find HITs. Type: String Default: None Constraints: The complete string of keywords, including commas and spaces, cannot be more than 1,000 characters.	No
`MaxAssignments`	The number of times the HIT can be accepted and completed before the HIT becomes unavailable. Type: positive integer Valid Values: any integer between 1 and 1000000000 (1 billion). Default: 1	No
`AutoApprovalDelayInSeconds`	The number of seconds after an assignment for the HIT has been submitted, after which the assignment is considered **Approved** automatically unless the Requester explicitly rejects it. Type: non-negative integer Valid Values: any integer between 0 (auto-approve results as soon as they are submitted) and 2592000 (30 days). Default: 2592000 (30 days)	No

Name	Description	Required
`QualificationRequirement`	A condition that a Worker's Qualifications must meet before the Worker is allowed to accept and complete the HIT. Type: a QualificationRequirement (p. 163) data structure. Default: None Constraints: there can be no more than 10 `QualificationRequirement` data structures for each HIT.	No
`AssignmentReviewPolicy`	The Assignment-level Review Policy applies to the assignments under the HIT. You can specify for Mechanical Turk to take various actions based on the policy. For more information, see Assignment Review Policies (p. 179). Type: String Valid Values: Must be a HIT Review Policy data structure, see HIT Review Policy (p. 151). Default: none	No
`HITReviewPolicy`	The HIT-level Review Policy applies to the HIT. You can specify for Mechanical Turk to take various actions based on the policy. For more information, see HIT Review Policies (p. 181). Type: String Valid Values: Must be a HIT Review Policy data structure, see HIT Review Policy (p. 151). Default: none	No
`RequesterAnnotation`	An arbitrary data field. The `RequesterAnnotation` parameter lets your application attach arbitrary data to the HIT for tracking purposes. For example, the `RequesterAnnotation` parameter could be an identifier internal to the Requester's application that corresponds with the HIT. Type: String Default: None Constraints: must not be longer than 255 characters in length.	No

Name	Description	Required
`UniqueRequestToken`	A unique identifier for this request. Allows you to retry the call on error without creating duplicate HITs. This is useful in cases such as network timeouts where it is unclear whether or not the call succeeded on the server. If the HIT already exists in the system from a previous call using the same `UniqueRequestToken`, subsequent calls will return a *AWS.MechanicalTurk.HitAlreadyExists* error with a message containing the HITId. Type: String Default: None Constraints: must not be longer than 64 characters in length. Note: It is your responsibility to ensure uniqueness of the token. The unique token expires after 24 hours. Subsequent calls using the same `UniqueRequestToken` made after the 24 hour limit could create duplicate HITs.	No

Response Elements

A successful request for the `CreateHIT` operation includes the elements described in the following table.

Name	Description
`HIT`	Contains the newly created HIT data. For a description of the HIT data structure as it appears in responses, see the HIT Data Structure (p. 144).

Examples

The following examples show how to use the `CreateHIT` operation.

Sample Request

The following are examples of REST requests.

Example Request (Query) Using CreateHIT with a HIT Type ID

The following example creates a simple HIT, using an explicit HIT type ID. The `Question` parameter takes a block of XML data as its value. See the QuestionForm (p. 191) data structure and the ExternalQuestion (p. 216) data structure for more information.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=CreateHIT
```

```
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITTypeId=T100CN9P324W00EXAMPLE
&Question=[URL-encoded question data]
&LifetimeInSeconds=604800
```

Example Request (Query) Using CreateHIT With a HIT Type ID Using SimplePlurality/2011-09-01 Review Policy

The following example creates a simple HIT with a SimplePlurality/2011-09-01 Review Policy.

```
<CreateHITRequest>
    <HITTypeId>T100CN9P324W00EXAMPLE</HITTypeId>
    <Question>[CDATA block or XML Entity encoded]</Question>
    <LifetimeInSeconds>604800</LifetimeInSeconds>
    <AssignmentReviewPolicy>
        <PolicyName>ScoreMyKnownAnswers/2011-09-01</PolicyName>
        <Parameter>
            <Key>AnswerKey</Key>
            <MapEntry>
                <Key>QuestionId3</Key>   <!—correct answer is “B” -->
                <Value>B</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId7</Key>  <!—correct answer is “A” -->
                <Value>A</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId15</Key>   <!—correct answer is “F” -->
                <Value>F</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId17</Key>   <!—correct answer is “C” -->
                <Value>C</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId18</Key>   <!—correct answer is “A” -->
                <Value>A</Value>
            </MapEntry>
        </Parameter>
        <Parameter>
           <Key>ExtendIfKnownAnswerScoreIsLessThan</Key>
           <Value>80</Value>
        </Parameter>
        <Parameter>
           <Key>ExtendMaximumAssignments</Key>
           <Value>3</Value>
        </Parameter>
    </AssignmentReviewPolicy>
</CreateHITRequest>
```

Example Request (Query) Using CreateHIT Without a HIT Type ID

The following example creates a simple HIT with some properties, letting Amazon Mechanical Turk determine the HIT type ID from the property values. The `Question` parameter takes a block of XML data as its value. See the QuestionForm (p. 191) data structure and the ExternalQuestion (p. 216) data structure for more information.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
```

```
&Operation=CreateHIT
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&Title=Location%20and%20Photograph%20Identification
&Description=Select%20the%20image%20that%20best%20represents...
&Reward.1.Amount=5
&Reward.1.CurrencyCode=USD
&Question=[URL-encoded question data]
&AssignmentDurationInSeconds=30
&LifetimeInSeconds=604800
&Keywords=location,%20photograph,%20image,%20identification,%20opinion
```

Sample Response

Amazon Mechanical Turk might return the following response for the preceding requests.

```
<CreateHITResponse>
  <OperationRequest>
    <RequestId>ece2785b-6292-4b12-a60e-4c34847a7916</RequestId>
  </OperationRequest>
  <HIT>
    <Request>
      <IsValid>True</IsValid>
    </Request>
    <HITId>GBHZVQX3EHXZ2AYDY2T0</HITId>
    <HITTypeId>NYVZTQ1QVKJZXCYZCZVZ</HITTypeId>
  </HIT>
</CreateHITResponse>
```

CreateQualificationType

Description

The `CreateQualificationType` operation creates a new Qualification type, which is represented by a QualificationType (p. 171) data structure.

Request Parameters

`CreateQualificationType` accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `CreateQualificationType` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: CreateQualificationType Default: None	Yes
`Name`	The name you give to the Qualification type. The type name is used to represent the Qualification to Workers, and to find the type using a Qualification type search. Type: String Default: None Constraints: Must be unique across all of your Qualification types.	Yes
`Description`	A long description for the Qualification type. On the Amazon Mechanical Turk website, the long description is displayed when a Worker examines a Qualification type. Type: String Default: None Constraints: Must be less than or equal to 2000 characters.	Yes
`Keywords`	One or more words or phrases that describe the Qualification type, separated by commas. The keywords of a type make the type easier to find during a search. Type: String Default: None	No

Name	Description	Required
	Constraints: Must be less than or equal to 1000 characters, including commas and spaces.	
`RetryDelayInSeconds`	The number of seconds that a Worker must wait after requesting a Qualification of the Qualification type before the worker can retry the Qualification request. Type: Non-negative integer Default: None. 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. It is not possible to disable retries for a Qualification type after it has been created with retries enabled. If you want to disable retries, you must dispose of the existing retry-enabled Qualification type using DisposeQualificationType (p. 40) and then create a new Qualification type with retries disabled.	No
`QualificationTypeStatus`	The initial status of the Qualification type. Type: String Valid Values: Active | Inactive Default: None	Yes
`Test`	The questions for the Qualification test a Worker must answer correctly to obtain a Qualification of this type. If this parameter is specified, `TestDurationInSeconds` must also be specified. Type: String Default: None. If not specified, the Worker may request the Qualification without answering any questions. Constraints: Must not be longer than 65535 bytes. Must be a QuestionForm (p. 191) data structure. This parameter cannot be specified if `AutoGranted` is `true`.	No
`AnswerKey`	The answers to the Qualification test specified in the `Test` parameter, in the form of an AnswerKey (p. 212) data structure. Type: String Default: None. If not specified, you must process Qualification requests manually. Constraints: Must not be longer than 65535 bytes.	No

Name	Description	Required
`TestDurationInSeconds`	The number of seconds the Worker has to complete the Qualification test, starting from the time the Worker requests the Qualification. Type: Integer Valid Values: Positive integer Default: None Conditions: Required if the `Test` parameter is specified.	Conditional
`AutoGranted`	Specifies whether requests for the Qualification type are granted immediately, without prompting the Worker with a Qualification test. Type: Boolean Valid Values: `true` | `false` Default: None Constraints: If the `Test` parameter is specified, this parameter cannot be `true`.	No
`AutoGrantedValue`	The Qualification value to use for automatically granted Qualifications. This parameter is used only if the `AutoGranted` parameter is `true`. Type: Integer Default: 1	No

Response Elements

A successful request for the `CreateQualificationType` operation includes the elements found in the following table:

Name	Description
`QualificationType`	The created Qualification type. Type: A QualificationType (p. 171) data structure.

Examples

The following example shows how to use the `CreateQualificationType` operation.

Sample Request

The following example creates a Qualification type.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=CreateQualificationType
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&Name=EnglishWritingAbility
&Description=The%20ability%20to%20write%20and%20edit%20text...
&QualificationTypeStatus=Active
```

Sample Response

The following is an example response.

```
<CreateQualificationTypeResponse>
  <OperationRequest>
    <RequestId>5218189c-1d7e-49a3-abbf-672fb5e77c66</RequestId>
  </OperationRequest>
  <QualificationType>
    <Request>
      <IsValid>True</IsValid>
    </Request>
    <QualificationTypeId>ZSPJXD4F1SFZP7YNJWR0</QualificationTypeId>
    <CreationTime>2009-07-13T17:26:33Z</CreationTime>
    <Name>SampleQualificationTest</Name>
    <Description>Description of my qualification test.</Description>
    <QualificationTypeStatus>Active</QualificationTypeStatus>
    <AutoGranted>0</AutoGranted>
  </QualificationType>
</CreateQualificationTypeResponse>
```

Related Operations

- AssignQualification (p. 14)
- DisposeQualificationType (p. 40)
- GetQualificationType (p. 77)
- UpdateQualificationType (p. 136)

DisableHIT

Description

The `DisableHIT` operation removes a HIT from the Amazon Mechanical Turk marketplace, approves any submitted assignments pending approval or rejection, and disposes of the HIT and all assignment data. Assignment results data cannot be retrieved for a HIT that has been disposed.

Assignments in progress at the time of the call to the `DisableHIT` operation are approved once the assignments are submitted. You will be charged for approval of these assignments.

When either all of the 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**. For more information about the **Reviewable** state, see Creating and Managing Assignments.

The `DisableHIT` operation does not work on HITs in the **Reviewable** state. For HITs in the **Reviewable** state, call the ApproveAssignment (p. 11) or the RejectAssignment (p. 111) operation for each submitted assignment for the HIT. Then call the DisposeHIT (p. 38) operation to dispose of the HIT.

Only the Requester who created the HIT can disable it.

Request Parameters

The `DisableHIT` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `DisableHIT` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: DisableHIT Default: None	Yes
`HITId`	The ID of the HIT, as returned by the CreateHIT (p. 21) operation. Type: String Default: None	Yes

Response Elements

A successful request for the `DisableHIT` operation returns with no errors. The response includes the elements in the following table. The operation returns no other data.

Name	Description
`DisableHITResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `DisableHIT` operation.

Sample Request

The following example disables a HIT with a specified HIT ID.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=DisableHIT
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITId=123RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<DisableHITResult>
  <Request>
  <IsValid>True</IsValid>
  </Request>
</DisableHITResult>
```

DisposeHIT

Description

The `DisposeHIT` operation disposes of a HIT that is no longer needed. Only the Requester who created the HIT can dispose of it.

You can only dispose of HITs that are in the **Reviewable** state, with all of their submitted assignments already either approved or rejected. If you call the `DisposeHIT` operation on a HIT that is not in the **Reviewable** state (for example, that has not expired, or still has active assignments), or on a HIT that is **Reviewable** but without all of its submitted assignments already approved or rejected, the service returns an error.

Notes

- HITs are automatically disposed of after 120 days.
- After you dispose of a HIT, you can no longer approve the HIT's rejected assignments.
- Disposed of HITs are not returned in results for the `SearchHITs` operation.
- Disposing of HITs can improve the performance of operations such as `GetReviewableHITs` and `SearchHITs`.

Request Parameters

The `DisposeHIT` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `DisposeHIT` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: DisposeHIT Default: None	Yes
`HitId`	The ID of the HIT, as returned by the CreateHIT (p. 21) operation. Type: String Default: None	Yes

Response Elements

A successful request for the `DisposeHIT` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
DisposeHITResult	Contains a Request element if the **Request** ResponseGroup is specified.

Examples

The following example shows how to use the DisposeHIT operation.

Sample Request

The following example disposes of the HIT with the specified HIT ID.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=DisposeHIT
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITId=123RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<DisposeHITResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</DisposeHITResult>
```

DisposeQualificationType

Description

The `DisposeQualificationType` operation disposes a Qualification type and disposes any HIT types that are associated with the Qualification type. A Qualification type is represented by a QualificationType (p. 171) data structure.

This operation does not revoke Qualifications already assigned to Workers because the Qualifications might be needed for active HITs. If there are any pending requests for the Qualification type, Amazon Mechanical Turk rejects those requests.

After you dispose of a Qualification type, you can no longer use it to create HITs or HIT types.

Note
`DisposeQualificationType` must wait for all the HITs that use the disposed Qualification type to be disposed before completing. It may take up to 48 hours before `DisposeQualificationType` completes and the unique name of the disposed Qualification type is available for reuse with CreateQualificationType (p. 32).

Request Parameters

A request to the Amazon Mechanical Turk Service includes parameters that control its behavior and the data it returns. Required parameters must be included for the request to succeed.

`DisposeQualificationType` accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `DisposeQualificationType` operation:

Name	Description	Required
`Operation`	The operation you want to call. To access the `DisposeQualificationType` operation, set the `Operation` parameter to **`DisposeQualificationType`**. Type: DisposeQualificationType Default: None	Yes
`QualificationTypeId`	The ID of the QualificationType (p. 171) to dispose. Type: String Default: None Constraint: A valid QualificationType (p. 171) ID.	Yes

Response Elements

A successful request for the `DisposeQualificationType` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`DisposeQualificationTypeResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `DisposeQualification` operation.

Sample Request

The following example disposes a Qualification type and any HIT types that are associated with the Qualification type.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=DisposeQualificationType
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&QualificationTypeId=AZ34EXAMPLE
```

Sample Response

The following is an example response.

```
<DisposeQualificationTypeResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</DisposeQualificationTypeResult>
```

Related Operations

- AssignQualification (p. 14)
- CreateQualificationType (p. 32)
- GetQualificationType (p. 77)
- UpdateQualificationType (p. 136)

ExtendHIT

Description

The `ExtendHIT` operation increases the maximum number of assignments, or extends the expiration date, of an existing HIT.

To extend the maximum number of assignments, specify the number of additional assignments.

To extend the expiration date, specify an amount of time as a number of seconds. If the HIT has not yet expired, the new expiration date is the existing date plus the amount of time specified. If the HIT has already expired, the new expiration date is the current time plus the amount of time specified.

Only the Requester who created a HIT can extend it.

Note

- HITs created with fewer than 10 assignments cannot be extended to have 10 or more assignments. Attempting to add assignments in a way that brings the total number of assignments for a HIT from fewer than 10 assignments to 10 or more assignments will result in an `AWS.MechanicalTurk.InvalidMaximumAssignmentsIncrease` exception.
- HITs that were created before July 22, 2015 cannot be extended. Attempting to extend HITs that were created before July 22, 2015 will result in an `AWS.MechanicalTurk.HITTooOldForExtension` exception.

Request Parameters

The `ExtendHIT` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `ExtendHIT` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: ExtendHIT Default: None	Yes
`HITId`	The ID of the HIT to extend Type: String Default: None	Yes
`MaxAssignmentsIncrement`	The number of assignments by which to increment the `MaxAssignments` parameter of the HIT. Type: positive integer Valid Values: any integer between 1 and 1000000000 (one billion)	No

Name	Description	Required
	Default: None	
`ExpirationIncrementInSeconds`	The amount of time, in seconds, by which to extend the expiration date. If the HIT has not yet expired, this amount is added to the HIT's expiration date. If the HIT has expired, the new expiration date is the current time plus this value. Type: positive integer Valid Values: any integer between 3600 (1 hour) and 31536000 (365 days) Default: None	No
`UniqueRequestToken`	A unique identifier for this request, which allows you to retry the call on error without extending the HIT multiple times. This is useful in cases such as network timeouts where it is unclear whether or not the call succeeded on the server. If the extend HIT already exists in the system from a previous call using the same `UniqueRequestToken`, subsequent calls will return an error with a message containing the request ID. Type: String Default: None Constraints: must not be longer than 64 characters in length. Note: It is your responsibility to ensure uniqueness of the token. The unique token expires after 24 hours. Subsequent calls using the same `UniqueRequestToken` made after the 24 hour limit could extend the HIT multiple times.	No

Response Elements

A successful request for the `ExtendHIT` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`ExtendHITResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `ExtendHIT` operation.

Sample Request

The following example extends the expiration date of a HIT by 5 days (432,000 seconds).

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=ExtendHIT
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITId=123RVWYBAZW00EXAMPLE
&ExpirationIncrementInSeconds=432000
```

Sample Response

The following is an example response.

```
<ExtendHITResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</ExtendHITResult>
```

ForceExpireHIT

Description

The `ForceExpireHIT` operation causes a HIT to expire immediately, as if the `LifetimeInSeconds` parameter of the HIT had elapsed.

The effect is identical to the HIT expiring on its own; the HIT no longer appears on the Amazon Amazon Mechanical Turk web site, and no new Workers are allowed to accept the HIT. Workers who have accepted the HIT prior to expiration are allowed to complete it or return it, or allow the assignment duration to elapse (abandon the HIT). Once all remaining assignments have been submitted, the expired HIT becomes **Reviewable**, and will be returned by a call to the GetReviewableHITs (p. 89) operation.

> **Note**
> Unlike the DisableHIT (p. 36) operation, the `ForceExpireHIT` operation does not have any effect on assignments. If assignments have been submitted for the HIT, your application still needs to approve or reject them before disposing of the HIT.

Request Parameters

The `ForceExpireHIT` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `ForceExpireHIT` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: ForceExpireHIT Default: None	Yes
`HITId`	The ID of the HIT, as returned by the `CreateHIT` operation. Type: String Default: None	Yes

Response Elements

A successful request for the `ForceExpireHIT` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`ForceExpireHITResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `ForceExpireHIT` operation.

Sample Request

The following example causes the specified HIT to expire.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Operation=ForceExpireHIT
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITId=123RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<ForceExpireHITResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</ForceExpireHITResult>
```

GetAccountBalance

Description

The `GetAccountBalance` operation retrieves the amount of money in your Amazon Mechanical Turk account.

Request Parameters

The `GetAccountBalance` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameter is specific to the `GetAccountBalance` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: GetAccountBalance Default: None	Yes

Response Elements

A successful request for the `GetAccountBalance` operation returns with a `GetAccountBalanceResult` element in the response.

The `GetAccountBalanceResult` element contains the following elements:

Name	Description
AvailableBalance	The amount available to pay for assignments. This is your current balance minus any outstanding payments, fees or bonuses you owe. Type: Price (p. 157) data structure
OnHoldBalance	Not used. This value is always 0. Type: Price (p. 157) data structure

Examples

The following example shows how to use the `GetAccountBalance` operation.

Sample Request

The following example retrieves the Requester's account balance.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
```

```
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetAccountBalance
&Signature=[signature for this request]
&Timestamp=[your system's local time]
```

Sample Response

The following is an example response.

```
<GetAccountBalanceResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <AvailableBalance>
    <Amount>10000.000</Amount>
    <CurrencyCode>USD</CurrencyCode>
    <FormattedPrice>$10,000.00</FormattedPrice>
  </AvailableBalance>
</GetAccountBalanceResult>
```

GetAssignment

Description

The `GetAssignment` operation retrieves an assignment with an `AssignmentStatus` value of `Submitted`, `Approved`, or `Rejected`, using the assignment's ID. Requesters can only retrieve their own assignments for HITs that they have not disposed of. For more information about the `AssignmentStatus` element, see the Assignment (p. 140) data structure. For more information about assignments, see Creating and Managing Assignments.

Request Parameters

The `GetAssignment` operation accepts parameters common to all operations. Some common parameters are required. For more information, see Common Parameters (p. 6).

The following parameters are specific to the `GetAssignment` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: GetAssignment Default: None	Yes
`AssignmentId`	The ID of the assignment that is being requested. Type: String Default: None.	Yes

Response Elements

A successful request for the `GetAssignment` operation has a `GetAssignmentResult` element in the response.

The `GetAssignmentResult` element contains the following elements:

Name	Description
`Request`	This element is present only if the **Request** `ResponseGroup` is specified.
`Assignment`	The assignment. The response includes one `Assignment` element. Type: An Assignment (p. 140) data structure
`HIT`	The HIT associated with this assignment. The response includes one `HIT` element. Type: A HIT (p. 144) data structure

Examples

The following example shows how to use the `GetAssignment` operation.

Sample Request

The following example retrieves an assignment using the assignment's ID.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetAssignment
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&AssignmentId=GYFTRHZ5J3DZREY48WNZE38ZR9RR1ZPMXGWE7WE0
```

Sample Response

The following is an example response.

```
<GetAssignmentResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <Assignment>
    <AssignmentId>GYFTRHZ5J3DZREY48WNZE38ZR9RR1ZPMXGWE7WE0</AssignmentId>
    <WorkerId>AD20WXZZP9XXK</WorkerId>
    <HITId>GYFTRHZ5J3DZREY48WNZ</HITId>
    <AssignmentStatus>Approved</AssignmentStatus>
    <AutoApprovalTime>2012-08-12T19:21:54Z</AutoApprovalTime>
    <AcceptTime>2012-07-13T19:21:40Z</AcceptTime>
    <SubmitTime>2012-07-13T19:21:54Z</SubmitTime>
    <ApprovalTime>2012-07-13T19:27:54Z</ApprovalTime>
    <Answer>
      <?xml version="1.0" encoding="UTF-8"?>
      <QuestionFormAnswers xmlns="http://mechanicalturk.amazonaws.com/
AWSMechanicalTurkDataSchemas/2005-10-01/QuestionFormAnswers.xsd">
        <Answer>
          <QuestionIdentifier>Question100</QuestionIdentifier>
          <FreeText>Move to X.</FreeText>
        </Answer>
      </QuestionFormAnswers>
    </Answer>
  </Assignment>
  <HIT>
    <HITId>GYFTRHZ5J3DZREY48WNZ</HITId>
    <HITTypeId>NYVZTQ1QVKJZXCYZCZVZ</HITTypeId>
    <CreationTime>2012-07-07T00:56:40Z</CreationTime>
    <Title>Location</Title>
    <Description>Answer this Question</Description>
    <Question>
      <QuestionForm xmlns="http://mechanicalturk.amazonaws.com/
AWSMechanicalTurkDataSchemas/2005-10-01/QuestionForm.xsd">
        <Question>
          <QuestionIdentifier>Question100</QuestionIdentifier>
          <DisplayName>My Question</DisplayName>
          <IsRequired>true</IsRequired>
          <QuestionContent>
            <Binary>
              <MimeType>
                <Type>image</Type>
```

```
              <SubType>gif</SubType>
            </MimeType>
            <DataURL>http://tictactoe.amazon.com/game/01523/board.gif</DataURL>
            <AltText>The game board, with "X" to move.</AltText>
          </Binary>
        </QuestionContent>
        <AnswerSpecification><FreeTextAnswer/></AnswerSpecification>
      </Question>
    </QuestionForm>
  </Question>
  <HITStatus>Assignable</HITStatus>
  <MaxAssignments>1</MaxAssignments>
  <Reward>
    <Amount>5.00</Amount>
    <CurrencyCode>USD</CurrencyCode>
    <FormattedPrice>$5.00</FormattedPrice>
  </Reward>
  <AutoApprovalDelayInSeconds>2592000</AutoApprovalDelayInSeconds>
  <Expiration>2012-07-14T00:56:40Z</Expiration>
  <AssignmentDurationInSeconds>30</AssignmentDurationInSeconds>
  <HITReviewStatus>NotReviewed</HITReviewStatus>
 </HIT>
</GetAssignment>
```

Related Operations

- GetAssignmentsForHIT (p. 52)

GetAssignmentsForHIT

Description

The `GetAssignmentsForHIT` operation retrieves completed assignments for a HIT. You can use this operation to retrieve the results for a HIT.

You can get assignments for a HIT at any time, even if the HIT is not yet **Reviewable**. If a HIT requested multiple assignments, and has received some results but has not yet become **Reviewable**, you can still retrieve the partial results with this operation.

Use the `AssignmentStatus` parameter to control which set of assignments for a HIT are returned. The `GetAssignmentsForHIT` operation can return submitted assignments awaiting approval, or it can return assignments that have already been approved or rejected. You can set `AssignmentStatus=Approved,Rejected` to get assignments that have already been approved and rejected together in one result set.

Only the Requester who created the HIT can retrieve the assignments for that HIT.

Results are sorted and divided into numbered pages and the operation returns a single page of results. You can use the parameters of the operation to control sorting and pagination.

Request Parameters

The `GetAssignmentsForHIT` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetAssignmentsForHIT` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: GetAssignmentsForHIT Default: None	Yes
`HITId`	The ID of the HIT for which completed assignments are requested. Type: String Default: None	Yes
`AssignmentStatus`	The status of the assignments to return. Type: String Valid Values: Submitted | Approved | Rejected Default: None. If not specified, the operation returns all assignments that have been submitted, including those that have been approved or rejected.	No

Name	Description	Required
	Note: Set `AssignmentStatus=Approved,Rejected` to get assignments that have been approved and rejected together in one result set.	
`SortProperty`	The field on which to sort the results returned by the operation. Type: String Valid Values: AcceptTime | SubmitTime | AssignmentStatus Default: SubmitTime	
`SortDirection`	The direction of the sort used with the field specified by the `SortProperty` parameter. Type: String Valid Values: Ascending | Descending Default: Ascending	No
`PageSize`	The number of assignments to include in a page of results. The complete sorted result set is divided into pages of this many assignments. Type: positive integer Valid Values: any integer between 1 and 100 Default: 10	No
`PageNumber`	The page of results to return. Once the assignments have been filtered, sorted, and divided into pages of size `PageSize`, the page corresponding to `PageNumber` is returned as the results of the operation. Type: positive integer Default: 1	No

Response Elements

A successful request for the `GetAssignmentsForHIT` operation has a `GetAssignmentsForHITResult` element in the response.

The `GetAssignmentsForHITResult` element contains the following elements:

Name	Description
NumResults	The number of assignments on the page in the filtered results list, equivalent to the number of assignments returned by this call. Type: non-negative integer
PageNumber	The number of the page in the filtered results list being returned.

Name	Description
	Type: positive integer
TotalNumResults	The total number of HITs in the filtered results list based on this call. Type: positive integer
Assignment	The assignment. The response includes one `Assignment` element for each HIT returned by the query. Type: an Assignment (p. 140) data structure

Examples

The following example shows how to use the `GetAssignmentsForHIT` operation.

Sample Request

The following example retrieves five assignments for a HIT, using the default sort order (`SubmitTime`) and direction (`Ascending`).

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetAssignmentsForHIT
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITId=123RVWYBAZW00EXAMPLE
&PageSize=5
&PageNumber=1
```

Sample Response

The following is an example response.

```
<GetAssignmentsForHITResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <NumResults>1</NumResults>
  <TotalNumResults>1</TotalNumResults>
  <PageNumber>1</PageNumber>
  <Assignment>
    <AssignmentId>GYFTRHZ5J3DZREY48WNZE38ZR9RR1ZPMXGWE7WE0</AssignmentId>
    <WorkerId>AD20WXZZP9XXK</WorkerId>
    <HITId>GYFTRHZ5J3DZREY48WNZ</HITId>
    <AssignmentStatus>Approved</AssignmentStatus>
    <AutoApprovalTime>2009-08-12T19:21:54Z</AutoApprovalTime>
    <AcceptTime>2009-07-13T19:21:40Z</AcceptTime>
    <SubmitTime>2009-07-13T19:21:54Z</SubmitTime>
    <ApprovalTime>2009-07-13T19:27:54Z</ApprovalTime>
    <Answer>
      <?xml version="1.0" encoding="UTF-8"?>
      <QuestionFormAnswers xmlns="http://mechanicalturk.amazonaws.com/
AWSMechanicalTurkDataSchemas/2005-10-01/QuestionFormAnswers.xsd">
        <Answer>
          <QuestionIdentifier>Question100</QuestionIdentifier>
```

```
          <FreeText>Move to X.</FreeText>
        </Answer>
      </QuestionFormAnswers>
    </Answer>
  </Assignment>
</GetAssignmentsForHITResult>
```

Related Operations

- GetAssignment (p. 49)
- SearchHITs (p. 117)

GetBlockedWorkers

Description

The `GetBlockedWorkers` operation retrieves a list of Workers who are blocked from working on your HITs.

Request Parameters

The `GetBlockedWorkers` operation accepts parameters that are common to all operations. Some common parameters are required. For more information, see Common Parameters (p. 6).

The following parameters are specific to the `GetBlockedWorkers` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: GetBlockedWorkers Default: None	Yes
`PageNumber`	The page of results to return. Once the assignments have been filtered, sorted, and divided into pages of size `PageSize`, the page corresponding to `PageSize` is returned as the results of the operation. Type: Positive integer Default: 1	No
`PageSize`	The number of assignments to include in a page of results. The complete sorted result set is divided into pages of this many assignments. Type: Positive integer Valid Values: Any integer between 1 and 65535 Default: 10	No

Response Elements

A successful request for the `GetBlockedWorkers` operation has a `GetBlockedWorkersResult` element in the response.

The `GetBlockedWorkersResult` element contains the elements described in the following table:

Name	Description
`Request`	This element is present only if the **Request** `ResponseGroup` is specified.

Name	Description
PageNumber	The number of the page in the filtered results list being returned. Type: Positive integer
NumResults	The number of assignments on the page in the filtered results list, equivalent to the number of assignments returned by this call. Type: Non-negative integer
TotalNumResults	The total number of HITs in the filtered results list based on this call. Type: Positive integer
WorkerBlock	The workers who have been blocked, along with the reason for the block. The response includes one WorkerBlock element for each worker. Type: A WorkerBlock (p. 177) data structure

Examples

The following example shows how to use the GetBlockedWorkers operation.

Sample Request

The following example blocks a Worker from working on your HITs.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetBlockedWorkers
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&PageNumber=1
&PageSize=10
```

Sample Response

The following is an example response.

```
<GetBlockedWorkersResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <PageNumber>1</PageNumber>
  <NumResults>2</NumResults>
  <TotalNumResults>2</TotalNumResults>
  <WorkerBlock>
    <WorkerId>A2QWESAMPLE1</WorkerId>
    <Reason>Poor quality work</Reason>
  </WorkerBlock>
  <WorkerBlock>
    <WorkerId>A2QWESAMPLE2</WorkerId>
    <Reason>Poor quality work</Reason>
  </WorkerBlock>
```

```
</GetBlockedWorkersResult>
```

Related Operations

To unblock a Worker, use the UnblockWorker (p. 132) operation.

GetBonusPayments

Description

The `GetBonusPayments` operation retrieves the amounts of bonuses you have paid to Workers for a given HIT or assignment.

Request Parameters

The `GetBonusPayments` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetBonusPayments` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: GetBonusPayments Default: None	Yes
`HITId`	The ID of the HIT associated with the bonus payments to retrieve. If not specified, all bonus payments for all assignments for the given HIT are returned. Type: String Default: None Conditions: Either the `HITId` parameter or the `AssignmentId` parameter must be specified.	Conditional
`AssignmentId`	The ID of the assignment associated with the bonus payments to retrieve. If specified, only bonus payments for the given assignment are returned. Type: String Default: None Conditions: Either the `HITId` parameter or the `AssignmentId` parameter must be specified.	Conditional
`PageSize`	The number of bonus payments to include in a page of results. The complete result set is divided into pages of this many bonus payments. Type: positive integer Valid Values: any integer between 1 and 100 Default: 10	No
`PageNumber`	The page of results to return. Once the list of bonus payments has been divided into pages of size `PageSize`,	No

Name	Description	Required
	the page corresponding to `PageNumber` is returned as the results of the operation. Type: positive integer Default: 1	

Response Elements

A successful request for the `GetBonusPayments` operation has a `GetBonusPaymentsResult` element in the response.

The `GetBonusPaymentsResult` element contains the following elements:

Name	Description
PageNumber	The page of results to return. Once the list of bonus payments has been divided into pages of size `PageSize`, the page corresponding to the `PageNumber` parameter is returned as the results of the operation. Type: positive integer
NumResults	The number of bonus payments on this page in the filtered results list, equivalent to the number of bonus payments being returned by this call. Type: non-negative integer
TotalNumResults	The total number of bonus payments in the filtered results list based on this call. Type: non-negative integer
BonusPayment	A bonus payment. The response includes one `BonusPayment` element for each bonus payment returned by the query. Type: A BonusPayment data structure, described in the next table.

Each `BonusPayment` is a data structure with the following elements:

Name	Description	
`WorkerId`	The ID of the Worker to whom the bonus was paid. Type: String	
`BonusAmount`	The amount of the bonus payment. Type: A Price (p. 157) data structure	
`AssignmentId`	The ID of the assignment associated with this bonus payment Type: String	
`Reason`	The **Reason** text given when the bonus was granted, if any.	

Name	Description	
	Type: String	
`GrantTime`	The date and time of when the bonus was granted. Type: A dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **`2005-01-31T23:59:59Z`**.	

Examples

The following example shows how to use the `GetBonusPayments` operation.

Sample Request

The following example retrieves all bonus payments associated with the specified HIT.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetBonusPayments
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITId=123RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<GetBonusPaymentsResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <NumResults>0</NumResults>
  <TotalNumResults>0</TotalNumResults>
  <PageNumber>1</PageNumber>
</GetBonusPaymentsResult>
```

GetFileUploadURL

Description

The `GetFileUploadURL` operation generates and returns a temporary URL. You use the temporary URL to retrieve a file uploaded by a Worker as an answer to a `FileUploadAnswer` question for a HIT. For information about the `FileUploadAnswer` answer, see QuestionForm (p. 191).

The temporary URL is generated the instant the `GetFileUploadURL` operation is called, and is valid for 60 seconds.

> **Note**
> URL expiration allows your application to retrieve the file without credentials, but still retain control over who can access your data, because you need an access key ID and signature to get the temporary URL. If you need to retrieve the file after the URL has expired, call `GetFileUploadURL` again to get a new URL.

You can get a temporary file upload URL any time until the HIT is disposed. After the HIT is disposed, any uploaded files are deleted, and cannot be retrieved.

Request Parameters

The `GetFileUploadURL` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetFileUploadURL` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: GetFileUploadURL Default: None	Yes
`AssignmentId`	The ID of the assignment that contains the question with a `FileUploadAnswer`. Type: String Default: None	Yes
`QuestionIdentifier`	The identifier of the question with a `FileUploadAnswer`, as specified in the QuestionForm (p. 191) of the HIT. Type: String Default: None	Yes

Response Elements

A successful request for the `GetFileUploadURL` operation has a `GetFileUploadURLResult` element in the response.

The `GetFileUploadURLResult` element includes the elements described in the following table.

Name	Description
`FileUploadURL`	A temporary URL for the file that the Worker uploaded for the answer. Type: URL

Examples

The following example shows how to use the `GetFileUploadURL` operation.

Sample Request

The following example of a call to the `GetFileUploadURL` operation retrieves the temporary URL for a file-upload answer to the given question in the given assignment.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetFileUploadURL
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&AssignmentId=123RVWYBAZW00EXAMPLE456RVWYBAZW00EXAMPLE
&QuestionIdentifier=ReadAloudAudio
```

Sample Response

The following is an example response.

```
<GetFileUploadURLResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <FileUploadURL>http://s3.amazonaws.com/myawsbucket/puppy.jpg</FileUploadURL>
</GetFileUploadURLResult>
```

GetHIT

Description

The `GetHIT` operation retrieves the details of the specified HIT.

Request Parameters

The `GetHIT` accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetHIT` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: GetHIT Default: None	Yes
`HITId`	The ID of the HIT to retrieve. Type: String Default: None	Yes

Response Elements

A successful request for the `GetHIT` operation returns the elements described in the following table in the response.

The `HIT` element contains the requested HIT data. For a description of the HIT data structure as it appears in responses, see the HIT (p. 144) data structure.

Name	Description
`HIT`	Contains the requested HIT data. Type: HIT Data Structure (p. 144)

Examples

The following example shows how to use the `GetHIT` operation.

Sample Request

The following example gets a HIT specified by a HIT ID.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
```

```
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetHIT
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITId=123RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<HIT>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <HITId>ZZRZPTY4ERDZWJ868JCZ</HITId>
  <HITTypeId>NYVZTQ1QVKJZXCYZCZVZ</HITTypeId>
  <CreationTime>2009-07-07T00:56:40Z</CreationTime>
  <Title>Location</Title>
  <Description>Select the image that best represents</Description>
  <Question>
    <QuestionForm xmlns="http://mechanicalturk.amazonaws.com/
AWSMechanicalTurkDataSchemas/2005-10-01/QuestionForm.xsd">
      <Question>
        <QuestionIdentifier>Question100</QuestionIdentifier>
        <DisplayName>My Question</DisplayName>
        <IsRequired>true</IsRequired>
        <QuestionContent>
          <Binary>
            <MimeType>
              <Type>image</Type>
              <SubType>gif</SubType>
            </MimeType>
            <DataURL>http://tictactoe.amazon.com/game/01523/board.gif</DataURL>
            <AltText>The game board, with "X" to move.</AltText>
          </Binary>
        </QuestionContent>
        <AnswerSpecification><FreeTextAnswer/></AnswerSpecification>
      </Question>
    </QuestionForm>
  </Question>
  <HITStatus>Assignable</HITStatus>
  <MaxAssignments>1</MaxAssignments>
  <Reward>
    <Amount>5.00</Amount>
    <CurrencyCode>USD</CurrencyCode>
    <FormattedPrice>$5.00</FormattedPrice>
  </Reward>
  <AutoApprovalDelayInSeconds>2592000</AutoApprovalDelayInSeconds>
  <Expiration>2009-07-14T00:56:40Z</Expiration>
  <AssignmentDurationInSeconds>30</AssignmentDurationInSeconds>
  <HITReviewStatus>NotReviewed</HITReviewStatus>
</HIT>
```

GetHITsForQualificationType

Description

The `GetHITsForQualificationType` operation returns the HITs that use the given Qualification type for a Qualification requirement.

The operation returns HITs of any status, except for HITs that have been disposed with the DisposeHIT (p. 38) operation.

This operation returns only HITs that you created.

> **Note**
> For reasons internal to the service, there may be a delay between when a HIT is created and when the HIT will be returned from a call to `GetHITsForQualificationType`.

The operation divides the results into numbered pages and returns a single page of results. You can control pagination with parameters to the operation.

Request Parameters

The `GetHITsForQualificationType` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetHITsForQualificationType` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: GetHITsForQualificationType Default: None	Yes
`QualificationTypeId`	The ID of the Qualification type to use when querying HITs, as returned by the CreateQualificationType (p. 32) operation. The operation returns HITs that require that a Worker have a Qualification of this type. Type: String Default: None	Yes
`PageSize`	The number of HITs to include in a page of results. The complete results set is divided into pages of this many HITs. Type: positive integer Valid Values: any integer between 1 and 100 Default: 10	No

Name	Description	Required
PageNumber	The page of results to return. After the HITs are divided into pages of size PageSize, the operation returns the page corresponding to the PageNumber. Type: positive integer Default: 1	No

Response Elements

A successful request for the GetHITsForQualificationType operation returns a GetHITsForQualificationTypeResult element in the response.

The GetHITsForQualificationTypeResult element contains the following elements:

Name	Description
NumResults	The number of HITs on this page in the filtered results list, equivalent to the number of HITs being returned by this call. Type: non-negative integer
PageNumber	The number of this page in the filtered results list. Type: positive integer
TotalNumResults	The total number of HITs in the filtered results list based on this call. Type: non-negative integer
HIT	The HIT. The response includes one HIT element for each HIT returned by the query. Type: HIT (p. 144) data structure.

Examples

The following example shows how to use the GetHITsForQualificationType operation.

Sample Request

The following example returns HITs that use the specified Qualification type for a Qualification requirement.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetHITsForQualificationType
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&QualificationTypeId=789RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<GetHITsForQualificationTypeResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <NumResults>1</NumResults>
  <TotalNumResults>1</TotalNumResults>
  <PageNumber>1</PageNumber>
  <HIT>
    <HITId>123RVWYBAZW00EXAMPLE</HITId>
    <HITTypeId>T100CN9P324W00EXAMPLE</HITTypeId>
    <CreationTime>2009-06-15T12:00:01</CreationTime>
    <HITStatus>Assignable</HITStatus>
    <MaxAssignments>5</MaxAssignments>
    <AutoApprovalDelayInSeconds>86400</AutoApprovalDelayInSeconds>
    <LifetimeInSeconds>86400</LifetimeInSeconds>
    <AssignmentDurationInSeconds>300</AssignmentDurationInSeconds>
    <Reward>
      <Amount>25</Amount>
      <CurrencyCode>USD</CurrencyCode>
      <FormattedPrice>$0.25</FormattedPrice>
    </Reward>
    <Title>Location and Photograph Identification</Title>
    <Description>Select the image that best represents...</Description>
    <Keywords>location, photograph, image, identification, opinion</Keywords>
    <Question>
      &lt;QuestionForm&gt;
      [XML-encoded Question data]
      &lt;/QuestionForm&gt;
    </Question>
    <QualificationRequirement>
      <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
      <Comparator>GreaterThan</Comparator>
      <Value>18</Value>
    </QualificationRequirement>
  <HITReviewStatus>NotReviewed</HITReviewStatus>
</HIT>
</GetHITsForQualificationTypeResult>
```

GetQualificationsForQualificationType

Description

The `GetQualificationsForQualificationType` operation returns all of the Qualifications granted to Workers for a given Qualification type.

This operations divides the results into numbered pages and returns a single page of results. You can control pagination with parameters to the operation.

Request Parameters

The `GetQualificationsForQualificationType` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetQualificationsForQualificationType` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: GetQualificationForQualificationType Default: None	Yes
`QualificationTypeId`	The ID of the Qualification type of the Qualifications to return. Type: String Default: None	Yes
`Status`	The status of the Qualifications to return. Type: String Valid Values: Granted | Revoked Default: Granted	No
`PageSize`	The number of Qualifications to include in a page of results. The operation divides the complete result set into pages of this many Qualifications. Type: positive integer Valid Values: any number between 1 and 100 Default: 10	No
`PageNumber`	The page of results to return. Once the operation divides the Qualifications into pages of size `PageSize`, it returns the page corresponding to `PageNumber`. Type: positive integer	No

Name	Description	Required
	Default: 1	

Response Elements

A successful request for the `GetQualificationsForQualificationType` operation returns a `GetQualificationsForQualificationTypeResult` element in the response.

The `GetQualificationsForQualificationTypeResult` element contains the following elements:

Name	Description
`PageNumber`	The page of results to return. Once the operation divides the Qualifications into pages of size `PageSize`, the operation returns the page corresponding to `PageNumber`. Type: positive integer
`NumResults`	The number of Qualifications on this page in the filtered results list, equivalent to the number of Qualifications being returned by this call. Type: non-negative integer
`TotalNumResults`	The total number of Qualifications in the filtered results list based on this call. Type: non-negative integer
`Qualification`	The Qualification. The response includes one `Qualification` element for each Qualification returned by the query. Type: a Qualification (p. 159) data structure.

Examples

The following example shows how to use the `GetQualificationsForQualificationType` operation.

Sample Request

The following example returns the Qualifications assigned to Workers for the given Qualification type.

```
https://mechanicalturk.amazonaws.com/onca/xml?
Service=AWSMechanicalTurkRequester
&Operation=GetQualificationsForQualificationType
&Version=2008-08-02
&AWSAccessKeyId=[the Requester's Access Key ID]
&Signature=[signature for this request]
&Timestamp=2009-07-15T01:21:28.186Z
&QualificationTypeId=ZSPJXD4F1SFZP7YNJWR0
```

Sample Response

The following is an example response.

```
<GetQualificationsForQualificationTypeResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <NumResults>1</NumResults>
  <TotalNumResults>1</TotalNumResults>
  <PageNumber>1</PageNumber>
    <QualificationRequest>
    <QualificationRequestId>789RVWYBAZW00EXAMPLE951RVWYBAZW00EXAMPLE
    </QualificationRequestId>
    <QualificationTypeId>ZSPJXD4F1SFZP7YNJWR0</QualificationTypeId>
      <SubjectId>AZ3456EXAMPLE</SubjectId>
      <Test>
        &lt;QuestionForm&gt;
        [XML-encoded question data]
        &lt;/QuestionForm&gt;
      </Test>
      <Answer>
        &lt;QuestionFormAnswers&gt;
        [XML-encoded answer data]
        &lt;/QuestionFormAnswers&gt;
      </Answer>
      <SubmitTime>2009-07-15T01:21:28.296Z</SubmitTime>
    </QualificationRequest>
</GetQualificationsForQualificationTypeResult>
```

GetQualificationRequests

Description

The `GetQualificationRequests` operation retrieves requests for Qualifications of a particular Qualification type. The owner of the Qualification type calls this operation to poll for pending requests, and grants Qualifications based on the requests using the GrantQualification (p. 103) operation.

The `GetQualificationRequests` operation returns only those Qualifications that require the type owner's attention. The operation does not return requests awaiting Qualification test answers and requests that have already been granted.

Only the owner of the Qualification type can retrieve its requests.

The operation sorts the results, divides them into numbered pages, and returns a single page of results. You can control sorting and pagination with parameters to the operation.

Request Parameters

The `GetQualificationRequests` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetQualificationRequests` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: GetQualificationRequests Default: None	Yes
`QualificationTypeId`	The ID of the Qualification type, as returned by the CreateQualificationType (p. 32) operation. Type: String Default: None. If not specified, all requests for all of your Qualification types are considered for the results.	No
`SortProperty`	The field on which to sort the returned results. Type: String Valid Values: QualificationTypeId | SubmitTime Default: SubmitTime	No
`SortDirection`	The direction of the sort. Type: String Valid Values: Ascending | Descending Default: Ascending	No

Name	Description	Required
PageSize	The number of Qualification requests to include in a page of results. The operation divides the complete sorted result set into pages of this many Qualification requests. Type: positive integer Valid Values: any number between 1 and 100 Default: 10	No
PageNumber	The page of results to return. When the operation has filtered the Qualification requests, sorted them, and divided them into pages of size PageSize, the operation returns the page corresponding to the PageNumber parameter. Type: positive integer Default: 1	No

Response Elements

A successful request for the GetQualificationRequests operation returns a GetQualificationRequestsResult element in the response.

The GetQualificationRequestsResult element contains the following elements:

Name	Description
NumResults	The number of Qualification requests on this page in the filtered results list, equivalent to the number of Qualification requests being returned by this call. Type: non-negative integer
PageNumber	The number of this page in the filtered results list. Type: positive integer
TotalNumResults	The total number of Qualification requests in the filtered results list based on this call. Type: non-negative integer
QualificationRequest	The Qualification request. The response includes one QualificationRequest element for each Qualification request returned by the query. Type: a QualificationRequest (p. 161) data structure.

Examples

The following example shows how to use the GetQualificationRequests operation.

Sample Request

The following example retrieves Qualification requests for a specified Qualification type.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetQualificationRequests
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&QualificationTypeId=789RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<GetQualificationRequestsResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <NumResults>1</NumResults>
  <TotalNumResults>1</TotalNumResults>
  <PageNumber>1</PageNumber>
  <QualificationRequest>
    <QualificationRequestId>789RVWYBAZW00EXAMPLE951RVWYBAZW00EXAMPLE
    </QualificationRequestId>
    <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
    <SubjectId>AZ3456EXAMPLE</SubjectId>
    <Test>
      &lt;QuestionForm&gt;
      [XML-encoded question data]
      &lt;/QuestionForm&gt;
    </Test>
    <Answer>
      &lt;QuestionFormAnswers&gt;
      [XML-encoded answer data]
      &lt;/QuestionFormAnswers&gt;
    </Answer>
    <SubmitTime>2005-12-01T23:59:59Z</SubmitTime>
  </QualificationRequest>
</GetQualificationRequestsResult>
```

GetQualificationScore

Description

The `GetQualificationScore` operation returns the value of a Worker's Qualification for a given Qualification type.

To get a Worker's Qualification, you must know the Worker's ID. The Worker's ID is included in the assignment data returned by the GetAssignmentsForHIT (p. 52) operation.

Only the owner of a Qualification type can query the value of a Worker's Qualification of that type.

Request Parameters

The `GetQualificationScore` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetQualificationScore` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: GetQualificationScore Default: None	Yes
`QualificationTypeId`	The ID of the Qualification type, as returned by the CreateQualificationType (p. 32) operation. Type: String Default: None	Yes
`SubjectId`	The ID of the Worker whose Qualification is being updated. Type: String Default: None	Yes

Response Elements

A successful request for the `GetQualificationScore` operation includes the elements described in the following table.

Name	Description
`Qualification`	For information about the contents of the `Qualification` element, see the Qualification (p. 159) data structure.

Examples

The following example shows how to use the `GetQualificationScore` operation.

Sample Request

The following example gets the value of a Qualification for a given user and Qualification type.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetQualificationScore
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&QualificationTypeId=789RVWYBAZW00EXAMPLE
&SubjectId=AZ3456EXAMPLE
```

Sample Response

The following is an example response.

```
<GetQualificationScoreResult>
  <Qualification>
    <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
    <SubjectId>AZ3456EXAMPLE</SubjectId>
    <GrantTime>2005-01-31T23:59:59Z</GrantTime>
    <IntegerValue>95</IntegerValue>
  </Qualification>
</GetQualificationScoreResult>
```

GetQualificationType

Description

The `GetQualificationType` operation retrieves information about a Qualification type using its ID.

Request Parameters

The `GetQualificationType` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetQualificationType` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: GetQualificationType Default: None	Yes
`QualificationTypeId`	The ID of the Qualification type, as returned by the CreateQualificationType (p. 32) operation. Type: String Default: None	Yes

Response Elements

A successful request for the `GetQualificationType` operation returns the elements described in the following table:

Name	Description
`QualificationType`	For information about the data structure of a Qualification type, see QualificationType (p. 171) data structure.

Examples

The following example shows how to use the `GetQualificationType` operation.

Sample Request

The following example gets a Qualification type by its ID.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
```

```
&Operation=GetQualificationType
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&QualificationTypeId=789RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<GetQualificationTypeResult>
  <QualificationType>
    <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
    <CreationTime>2005-01-31T23:59:59Z</CreationTime>
    <Name>EnglishWritingAbility</Name>
    <Description>The ability to write and edit text...</Description>
    <Keywords>English, text, write, edit, language</Keywords>
    <QualificationTypeStatus>Active</QualificationTypeStatus>
    <RetryDelayInSeconds>86400</RetryDelayInSeconds>
    <IsRequestable>true</IsRequestable>
  </QualificationType>
</GetQualificationTypeResult>
```

GetRequesterStatistic

Description

The `GetRequesterStatistic` operation retrieves statistics about you (the Requester calling the operation). The following table describes the available statistics:

Name	Description
NumberAssignmentsAvailable	DEPRECATED* as of 11/01/2016 - The number of times Workers can accept an available HIT, totaled over all available HITs. In other words, a HIT with 3 `MaxAssignments` can be described as having 3 available assignments, each of become **Accepted** when a Worker accepts the HIT. (Technically, Amazon Mechanical Turk does not create an assignment with an assignment ID until a Worker accepts a HIT.) Type: Long
NumberAssignmentsAccepted	DEPRECATED* as of 11/01/2016 - The number of times Workers have accepted your HITs. Type: Long
NumberAssignmentsPending	The total number of assignments for your HITs that have been submitted by Workers and are awaiting approval. The total increases and decreases as assignments are submitted by Workers and approved or rejected by you. Type: Long
NumberAssignmentsApproved	The number of assignments you have approved. Type: Long
NumberAssignmentsRejected	The number of assignments you have rejected. Type: Long
NumberAssignmentsReturned	DEPRECATED* as of 11/01/2016 - The number of times Workers have returned assignments for your HITs. Type: Long
NumberAssignmentsAbandoned	The number of times Workers have abandoned assignments (allowed the deadline to elapse without submitting results) for your HITs. Type: Long
PercentAssignmentsApproved	The percentage of assignments that you have approved, computed over all assignments that you have approved or rejected. The percentage is represented as a decimal fraction between 0 and 1. The statistic value for a given day represents a change in the overall percentage due to activity for that day. Type: Double

Name	Description
PercentAssignmentsRejected	The percentage of assignments that you have rejected, computed over all assignments that you have approved or rejected. The percentage is represented as a decimal fraction between 0 and 1. The statistic value for a given day represents a change in the overall percentage due to activity for that day. Type: Double
TotalRewardPayout	The total amount of the rewards paid for approved assignments. The amount is given in U.S. dollars. Type: Double
AverageRewardAmount	The change in the average amount of the rewards paid for approved assignments. The amount is given in U.S. dollars. Type: Double
TotalRewardFeePayout	The total amount of the HIT listing fees paid for approved assignments. The amount is given in U.S. dollars. Type: Double
TotalFeePayout	The total amount of the HIT listing fees paid for approved assignments and bonus payments. The amount is given in U.S. dollars. *This statistic is deprecated.* To get the total amount of fees paid for rewards and bonuses, get the `TotalRewardFeePayout` statistic and the `TotalBonusFeePayout` statistic and add them together. Type: Double
TotalRewardAndFeePayout	The total amount of money paid for approved assignments, including rewards and fees. The amount is given in U.S. dollars. This total does *not* include fees for bonus payments made with the GrantBonus (p. 100) operation. *This statistic is deprecated.* To get the total amount of money paid for rewards and reward fees, get the `TotalRewardPayout` and `TotalRewardFeePayout` statistics and add them together. Type: Double
TotalBonusPayout	The total amount of the bonuses paid to Workers. The amount is given in U.S. dollars. Type: Double
TotalBonusFeePayout	The total amount of the fees paid for bonus payments. The amount is given in U.S. dollars. Type: Double

Name	Description
NumberHITsCreated	The number of HITs you created. Type: Long
NumberHITsCompleted	The total number of your HITs that have been completed to their final state of either **Disposed** or **Disabled**. Type: Long
NumberHITsAssignable	The number of your HITs with status **Assignable**. **Note** `NumberHITsAssignable` can only be queried as a **LifeToDate** value. While most statistics change in real time, a day's value for this statistic is added to the `LifeToDate` total at the end of the day. Type: Long
NumberHITsReviewable	The number of your HITs with status **Reviewable**. Type: Long
EstimatedRewardLiability	The total amount of all of the rewards for HITs and assignments that have not yet been completed. This includes the reward for each unclaimed assignment for HITs that have not yet expired, each assignment in progress, and each submitted assignment that has not yet been approved or rejected. This is an estimate, because it is possible that not all of a HIT's assignments will be completed before the HIT expires. The amount is given in U.S. dollars. Type: Double
EstimatedFeeLiability	The total amount of all of the HIT listing fees for HITs and assignments that have not yet been completed at a given point in time. The amount is given in U.S. dollars. Type: Double
EstimatedTotalLiability	The total amount of all of the rewards and fees for HITs and assignments that have not yet been completed at a given point in time. The amount is given in U.S. dollars. Type: Double

*Deprecated statistics are not expected to be accurate. The `GetRequesterStatistic` operation may return an exception when a deprecated metric is requested.

Request Parameters

The `GetRequesterStatistic` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetRequesterStatistic` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: GetRequesterStatistic Default: None	Yes
`Statistic`	The statistic to return Type: String Valid Values: See the preceding table. Default: None	Yes
`TimePeriod`	The time period of the statistic to return. Type: String Valid Values: OneDay | SevenDays | ThirtyDays | LifeToDate Default: None	Yes
`Count`	The number of data points to return Type: positive integer Default: 1 Conditions: only used if `TimePeriod` is `OneDay` For example, if `TimePeriod` is `OneDay` and `Count` is `12`, the operation returns 12 data points for the statistic, one for each of 12 calendar days leading up to the current date, including the current day.	Conditional

Response Elements

A successful request for the `GetRequesterStatistic` operation has a `GetStatisticResult` element in the response.

The `GetStatisticResult` element contains a the following elements for each value requested.

Name	Description
`Statistic`	The named statistic you specified in the Request. See the preceding table for a list of statistics. Type: String
`TimePeriod`	The time period you specified in the Request. Type: String

Name	Description
DataPoint	The data point data structure described in the next table. Type: DataPoint structure

Each DataPoint element contains the following elements:

Name	Description
Date	The date represented by the data point. For aggregate values, this is the current date. Type: A dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as 2005-01-31T23:59:59Z
LongValue | DoubleValue	The value of the statistic over the specified time period. The element name and data type depend on which statistic was requested. Type: a long or a double, depending on the requested statistic.

Examples

The following example shows how to use the GetRequesterStatistic operation.

Sample Request

The following example of a call to the GetRequesterStatistic operation retrieves the total reward payout for the thirty days leading up to the current date.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2011-09-01
&Operation=GetRequesterStatistic
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&Statistic=NumberAssignmentsApproved
&TimePeriod=ThirtyDays
&Count=1
```

Sample Response

The following is an example response.

```
<GetStatisticResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <Statistic>NumberAssignmentsApproved</Statistic>
  <TimePeriod>ThirtyDays</TimePeriod>
  <DataPoint>
    <Date>2011-09-05T07:00:00Z</Date>
    <DoubleValue>281</DoubleValue>
```

```
  </DataPoint>
</GetStatisticResult>
```

GetRequesterWorkerStatistic

Description

The `GetRequesterWorkerStatistic` operation retrieves statistics about a specific Worker who has completed Human Intelligence Tasks (HITs) for you. If you have used Review Policies with known answers or plurality, Mechanical Turk will summarize the following statistics about the Worker's known answers and agreement level. These statistics are only for your Requester account. For more information about Review Policies, see Review Policies (p. 178).

The following table describes the available statistics:

Name	Description
NumberAssignmentsApproved	The number of assignments you have approved for the Worker. Type: Long
NumberAssignmentsRejected	The number of assignments you have rejected for the Worker. Type: Long
PercentAssignmentsApproved	The percentage of assignments approved, which is the Number of assignments approved divided by the number of assignments approved or rejected. Type: Double
PercentAssignmentsRejected	The percentage of assignments rejected, which is the Number of assignments rejected divided by the number of assignments approved or rejected. Type: Double
NumberKnownAnswersCorrect	The total number of *known answer* questions that the Worker has answered correctly. Type: Long
NumberKnownAnswersIncorrect	The total number of *known answer* questions that the Worker has answered incorrectly. Type: Long
NumberKnownAnswersEvaluated	The total number of *known answer* questions in assignments the Worker has submitted. Type: Long
PercentKnownAnswersCorrect	The rounded percentage of *known answer* questions the Worker has answered correctly, which is the number of correct known answers divided by the number of known answers evaluated. Type: Double
NumberPluralityAnswersCorrect	The number of evaluated questions that the Worker provided the agreed-upon answer for.

Name	Description
	Type: Long
NumberPluralityAnswersIncorrect	The number of evaluated questions that the Worker did not provide the agreed-upon answer for. Type: Long
NumberPluralityAnswersEvaluated	The number of evaluated questions answered by the Worker participating in the HIT. Type: Long
PercentPluralityAnswersCorrect	The number of questions that the Worker provided the agreed-upon answer for, divided by the number of evaluated questions. Type: Double

Request Parameters

The `GetRequesterWorkerStatistic` operation accepts parameters common to all operations. Some common parameters are required. For more information, see Common Parameters (p. 6).

The following parameters are specific to the `GetRequesterWorkerStatistic` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: GetRequesterWorkerStatistic Default: None	Yes
`Statistic`	The statistic to return. Type: String Valid Values: See the preceding available statistics table. Default: None	Yes
`WorkerId`	The Worker you want to return the statistics for. Type: String Default: None	Yes
`TimePeriod`	The time period of the statistic to return. Type: String Valid Values: OneDay | SevenDays | ThirtyDays | LifeToDate Default: None	Yes

Name	Description	Required
`Count`	The number of data points to return. Type: Positive Integer Default: 1 Conditions: only used if `TimePeriod` is `OneDay`. For example, if `TimePeriod` is `OneDay` and `Count` is `12`, the operation returns 12 data points for the statistic, one for each of 12 calendar days leading up to the current date, including the current day.	Conditional

Response Elements

A successful request for the `GetRequesterWorkerStatistic` operation has a `GetStatisticResult` element in the response.

The `GetStatisticResult` element contains the elements in the following table for each value requested.

Name	Description
`WorkerId`	The Worker ID you are requesting the statistics for. Type: String
`Statistic`	The named statistic you specified in the Request. See the preceding table for a list of statistics. Type: String
`TimePeriod`	The time period you specified in the Request. Type: String
`DataPoint`	The data point data structure described in the next table. Type: DataPoint structure

Each `DataPoint` data structure contains the following elements:

Name	Description
`Date`	The date represented by the data point. For aggregate values, this is the current date. Type: A dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as `2005-01-31T23:59:59Z`
`LongValue` | `DoubleValue`	The value of the statistic over the specified time period. The element name and data type depend on which statistic was requested.

Name	Description
	Type: A Long or a Double, depending on the requested statistic.

Examples

The following example shows how to use the `GetRequesterWorkerStatistic` operation.

Sample Request

The following GetRequesterWorkerStatistic operation request retrieves the number of assignments approved for the Worker ID A1Z4X5D207ALZF in the last 30 days.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2011-09-01
&Operation=GetRequesterWorkerStatistic
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&Statistic=NumberAssignmentsApproved
&WorkerId=A1Z4X5D207ALZF
&TimePeriod=ThirtyDays
&Count=1
```

Sample Response

The following is an example response where the Worker had 281 assignments approved in the last 30 days.

```
<GetStatisticResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <WorkerId>A1Z4X5D207ALZF</WorkerId>
  <Statistic>NumberAssignmentsApproved</Statistic>
  <TimePeriod>ThirtyDays</TimePeriod>
  <DataPoint>
    <Date>2011-09-05T07:00:00Z</Date>
    <DoubleValue>281</DoubleValue>
  </DataPoint>
</GetStatisticResult>
```

GetReviewableHITs

Description

The `GetReviewableHITs` operation retrieves the HITs with `Status` equal to **Reviewable** or `Status` equal to **Reviewing** that belong to the Requester calling the operation.

You can limit the query to HITs with a specified HIT type.

The operation sorts the results, divides them into numbered pages, and returns a single page of results. You can control sorting and pagination can be controlled with parameters to the operation.

When (`PageNumber` x `PageSize`) is less than 100, you can get reliable results when you use any of the sort properties. If this number is greater than 100, use the **`Enumeration`** sort property for best results. The **`Enumeration`** sort property guarantees that the operation returns all reviewable HITs with no duplicates, but not in any specific order.

Request Parameters

The `GetReviewableHITs` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GetReviewableHITs` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: GetReviewableHITs Default: None	Yes
`HITTypeId`	The ID of the HIT type of the HITs to consider for the query. Type: String Default: None. If not specified, all of the Requester's HITs are considered for the query.	No
`Status`	The status of the HITs to return Type: String Valid Values: Reviewable | Reviewing Default: Reviewable To query both **Reviewable** and **Reviewing** HITs, specify multiple `Status` parameters.	No
`SortProperty`	The field on which to sort the results. Type: String Valid Values: Title | Reward | Expiration | CreationTime | Enumeration	No

Name	Description	Required
	Default: Expiration	
`SortDirection`	The direction of the sort used with the field specified by the `SortProperty` property. Type: String Valid Values: Ascending | Descending Default: Descending	No
`PageSize`	The number of HITs to include in a page of results. The operation divides the complete sorted result set is divided into pages of this many HITs. Type: positive integer Valid Values: any number between 1 and 100 Default: 10	No
`PageNumber`	The page of results to return. After the operation filters, sorts, and divides the HITs into pages of size `PageSize`, it returns the page corresponding to `PageNumber` as the results of the operation. Type: positive integer Default: 1	No

Response Elements

A successful request for the `GetReviewableHITs` operation has a `GetReviewableHITsResult` element in the response.

The `GetReviewableHITsResult` element contains the following elements:

Name	Description
`NumResults`	The number of HITs on this page in the filtered results list, equivalent to the number of HITs this call returns. Type: non-negative integer
`PageNumber`	The number of this page in the filtered results list. Type: positive integer
`TotalNumResults`	The total number of HITs in the filtered results list based on this call. Type: non-negative integer
`HIT`	The HIT. The response includes one `HIT` element for each HIT returned by the query. Type: a HIT (p. 144) data structure

Examples

The following example shows how to use the `GetReviewableHITs` operation.

Sample Request

The following example retrieves five of the Requester's reviewable HITs, using the default values for the `SortProperty` and `SortOrder` parameters (ExpirationDate, Ascending).

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GetReviewableHITs
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&PageSize=5
&PageNumber=1
```

Sample Response

The following is an example response.

```
<GetReviewableHITsResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <NumResults>1</NumResults>
  <TotalNumResults>1</TotalNumResults>
  <PageNumber>1</PageNumber>
  <HIT>
    <HITId>GBHZVQX3EHXZ2AYDY2T0</HITId>
  </HIT>
</GetReviewableHITsResult>
```

GetReviewResultsForHIT

Description

The `GetReviewResultsForHIT` operation retrieves the computed results and the actions taken in the course of executing your Review Policies during a CreateHIT (p. 21) operation. For information about how to apply Review Policies when you call `CreateHIT`, see Review Policies (p. 178). The `GetReviewResultsForHIT` operation can return results for both Assignment-level and HIT-level review results. You can also specify to only return results pertaining to a particular Assignment.

Request Parameters

The `GetReviewResultsForHIT` operation accepts parameters common to all operations. Some common parameters are required. For more information, see Common Parameters (p. 6).

The following parameters are specific to the `GetReviewResultsForHIT` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: GetReveiwResultsForHIT Default: None	Yes
`HITId`	The unique identifier of the HIT to retrieve review results for. Type: String Default: None	Yes
`PolicyLevel`	The Policy Level(s) to retrieve review results for. Type: String Default: HIT and Assignments Constraints: HIT and Assignment. If omitted, the default behavior is to retrieve all data for both policy levels. For a list of all the described policies, see Review Policies (p. 178).	No
`AssignmentId`	If supplied, the results are limited to those pertaining directly to this Assignment ID. Type: String Default: None Note: Both the HIT-level and the Assignment-level Review Policies may take actions and return results concerning an Assignment, so both will be reported unless filtered by the `PolicyLevel` parameter.	No

Name	Description	Required
RetrieveActions	Retrieves a list of the actions taken executing the Review Policies and their outcomes. Type: String Default: T Constraints: T or F	No
RetrieveResults	Retrieves a list of the results computed by the Review Policies. Type: String Default: T Constraints: T or F	No
PageSize	The number of results to include in a page of results. The complete results set is divided into pages of this many HITs. Type: positive integer Valid Values: any integer between 100 and 65535 Default: 400	Optional
PageNumber	The page of results to return. After the results are divided into pages of size PageSize, the operation returns the page corresponding to the PageNumber. Type: positive integer Default: 1	Optional

Response Elements

A successful request for the GetReviewResultsForHIT operation has a GetReviewResultsForHITResult element in the response. The GetReviewResultsForHITResult element contains the name of the Review Policy applied as well as the AssignmentReviewReport element and the HITReviewReport element. Additional elements called ReviewAction and ReviewResult provide the results of Review Policy specified when the HIT was created.

The GetReviewResultsForHITResult element contains the following elements:

Name	Description
HITId	The HIT ID. Type: String
AssignmentReviewPolicy	The name of the Assignment-level Review Policy. This contains only the PolicyName element. Type: String

Name	Description
	Constraint: Name of the applied policy. For a list of available named policies, see Review Policies (p. 178).
`HITReviewPolicy`	The name of the HIT-level Review Policy. This contains only the `PolicyName` element. Type: String Constraint: Name of the applied policy. For a list of available named policies, see Review Policies (p. 178).
`AssignmentReviewReport`	Contains both `ReviewResult` and `ReviewAction` elements for an Assignment. Type: `ReviewResult` data structure and `ReviewAction` data structure. See below for details.
`HITReviewReport`	Contains both`ReviewResult` and `ReviewAction` elements for a particular HIT. Type: `ReviewResult` data structure and `ReviewAction` data structure. See below for details.

ReviewResult Data Structure

Both the `AssignmentReviewReport` element and the`HITReviewReport` element contain the ReviewResult data structure. The ReviewResult structure is returned multiple times for each result specified in the Review Policy.

> **Note**
> A HIT-level Review Policy may be enacted multiple times if the HIT is extended by the Review Policy or by extending a reviewable HIT using the ExtendHIT (p. 42) operation.

`GetReviewResultsForHITResult` includes the results from each of the review steps, in chronological order. The first-produced results appear first in the list, while other results with the same SubjectId/QuestionId/Key may appear later in the list, reflecting later reviews. Assignment-level review policies are enabled precisely once per submitted Assignment, so the Assignment-level results will only contain duplicate SubjectId/QuestionId/Key elements if the Assignment-level policy contains information about the HIT itself.

ReviewResult Elements

The `ReviewResult` data structure contains the following elements:

Name	Description
`ActionId`	A unique identifier of the Review action result. Type: String
`SubjectId`	The HITID or AssignmentId about which this result was taken. Note that HIT-level Review Policies will often emit results about both the HIT itself and its Assignments, while Assignment-level review policies generally only emit results about the Assignment itself.

Name	Description
	Type: String
`ObjectType`	The type of the object from the SubjectId field. Type: String Constraint: Assignment or HIT
`QuestionId`	Specifies the QuestionId the result is describing. Depending on whether the ObjectType is a HIT or Assignment this results could specify multiple values. See the next table for possible values and their meaning. If ObjectType is HIT and QuestionId is absent, then the result describes results of the HIT, including the HIT agreement score. If ObjectType is Assignment and QuestionId is absent, then the result describes the Worker's performance on the HIT. Type: String
`Key`	Key identifies the particular piece of reviewed information. See the next table for details. Type: String
`Value`	The values of *Key* provided by the review policies you have selected. Type: String

Result Keys

The `QuestionId` element can contain the following result values:

Key	SubjectType	Meaning
AgreedAnswerFound	Question	A boolean to describe whether an agreed answer was found. Constraints: true or false
AgreedAnswer	Question	If AgreedAnswerFound is true, contains the agreed answer for the question. If no agreed answer is found then this value is not returned.
AnswerAgreementScore	Question	If AgreedAnswerFound is true, contains the Agreement Score for the question. If no agreed answer is found then this value is not returned.
WorkerAgreementScore	Assignment	Contains the computed Worker Agreement Score for the Worker that completed the assignment.
PluralityAnswersCorrect	Assignment	Contains the number of answers in the assignment that agreed with the majority.
PluralityAnswersIncorrect	Assignment	Contains the number of answers in the assignment that disagreed with the majority.
HitAgreementScore	HIT	Contains the computed HIT Agreement Score for the HIT.

ReviewAction Data Structure

Both the `AssignmentReviewReport` and the `HITReviewReport` elements contains the `ReviewAction` data structure. This structure is returned multiple times for each action specified in the Review Policy.

ReviewAction Elements

The `ReviewAction` data structure element contains the following elements:

Name	Description
`ActionId`	The unique identifier for the action. Type: String
`ActionName`	The nature of the action itself. The Review Policy is responsible for examining the HIT and Assignments, emitting results, and deciding which other actions will be necessary. Type: String Constraints: Approve, Reject, or Extend
`ObjectId`	The specific HITId or AssignmentID targeted by the action. Type: String Constraint: none
`ObjectType`	The type of object in ObjectId. Type: String Constraint: AssignmentID or HITID
`Status`	The current disposition of the action. See below for a description of the statuses. Type: String Constraints: INTENDED, SUCCEEDED, FAILED, or CANCELLED
`CompleteTime`	The date when the action was completed. Type: Date/Time Constraints: none
`Results`	A description of the outcome of the review. **Note** Do not to make assumptions about the format of this string or the data it contains. Type: String
`ErrorCode`	Present only when the Results have a FAILED Status. Type: String

ReviewAction Elements Returned Status Values

The following table lists the types of Status values returned in the `ReviewAction` Status element:

Status	Description
INTENDED	The action is either queued for execution or currently being worked on.
SUCCEEDED	The action was taken successfully. Note that the SUCCESS of an action does not necessarily imply that the object has actually been acted upon. For example, an action trying to approve an already-approved Assignment will report itself as SUCCESSFUL even though the Assignment was not further altered by the second approval. Similarly, if the policy was configured to extend a HIT to a maximum of 5 assignments, then an extend action attempted when the HIT has already been extended to have a maximum of 5 assignments will take no action and report SUCCESS, because it has successfully performed the action in accordance with the configured limits of the Review Policy.
FAILED	The action could not be taken because of the state of its object. For example, an Assignment that is already approved cannot later be rejected or a HIT cannot be extended if the Requester lacks sufficient funds to cover the increased liability. Whenever an action is "FAILED", an additional ErrorCode element is present.
CANCELLED	The action was prevented by the intervention of Mechanical Turk.

Examples

The following example shows how to use the `GetReviewResultsForHIT` operation.

Sample Request

The following is an example request.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2011-XX-XX
&Operation=GetReviewResultsForHIT
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITId=1AAAAAAAAABBBBBBBBBBCCCCCCCCCC
```

Sample Response

The following is an example response.

```
<GetReviewResultsForHITResult>
  <HITId>1AAAAAAAAABBBBBBBBBBCCCCCCCCCC</HITId>
  <AssignmentReviewPolicy>
    <PolicyName>ScoreMyKnownAnswers/2011-09-01</PolicyName>
  </AssignmentReviewPolicy>
  <HITReviewPolicy>
    <PolicyName>SimplePlurality/2011-09-01</PolicyName>
  </HITReviewPolicy>
```

```
<AssignmentReviewReport>
  <ReviewResult>
    <SubjectId>1DDDDDDDDDEEEEEEEEEEFFFFFFFFFF</SubjectId>
    <ObjectType>Assignment</ObjectType>
    <QuestionId>Question_2</QuestionId>
    <Key>KnownAnswerCorrect</Key>
    <Value>1</Value>
  </ReviewResult>
  <ReviewResult>
    <SubjectId>1DDDDDDDDDEEEEEEEEEEFFFFFFFFFF</SubjectId>
    <ObjectType>Assignment</ObjectType>
    <Key>KnownAnswerScore</Key>
    <Value>100</Value>
  </ReviewResult>
  <ReviewResult>
    <SubjectId>1GGGGGGGGGHHHHHHHHHHIIIIIIIIII</SubjectId>
    <ObjectType>Assignment</ObjectType>
    <QuestionId>Question_2</QuestionId>
    <Key>KnownAnswerCorrect</Key>
    <Value>0</Value>
  </ReviewResult>
  <ReviewResult>
    <SubjectId>1GGGGGGGGGHHHHHHHHHHIIIIIIIIII</SubjectId>
    <ObjectType>Assignment</ObjectType>
    <Key>KnownAnswerScore</Key>
    <Value>0</Value>
  </ReviewResult>
  <ReviewAction>
    <ActionName>review</ActionName>
    <ObjectId>1DDDDDDDDDEEEEEEEEEEFFFFFFFFFF</ObjectId>
    <ObjectType>Assignment</ObjectType>
    <Status>SUCCEEDED</Status>
    <Result>Reviewed one known answer; 1/1 correct</Result>
  </ReviewAction>
  <ReviewAction>
    <ActionName>approve</ActionName>
    <ObjectId>1DDDDDDDDDEEEEEEEEEEFFFFFFFFFF</ObjectId>
    <ObjectType>Assignment</ObjectType>
    <Status>SUCCEEDED</Status>
    <Result>Approved</Result>
  </ReviewAction>
  <ReviewAction>
    <ActionName>review</ActionName>
    <ObjectId>1GGGGGGGGGHHHHHHHHHHIIIIIIIIII</ObjectId>
    <ObjectType>Assignment</ObjectType>
    <Status>SUCCEEDED</Status>
    <Result>Reviewed one known answer; 0/1 correct</Result>
  </ReviewAction>
  <ReviewAction>
    <ActionName>reject</ActionName>
    <ObjectId>1GGGGGGGGGHHHHHHHHHHIIIIIIIIII</ObjectId>
    <ObjectType>Assignment</ObjectType>
    <Status>SUCCEEDED</Status>
    <Result>Rejected</Result>
  </ReviewAction>
</AssignmentReviewReport>
<HITReviewReport>
  <ReviewResult>
    <SubjectId>1DDDDDDDDDEEEEEEEEEEFFFFFFFFFF</SubjectId>
    <ObjectType>Assignment</ObjectType>
    <QuestionId>Question_1</QuestionId>
    <Key>AgreedWithPlurality</Key>
    <Value>1</Value>
  </ReviewResult>
  <ReviewResult>
    <SubjectId>1GGGGGGGGGHHHHHHHHHHIIIIIIIIII</SubjectId>
```

```
      <ObjectType>Assignment</ObjectType>
      <QuestionId>Question_1</QuestionId>
      <Key>AgreedWithPlurality</Key>
      <Value>1</Value>
    </ReviewResult>
    <ReviewResult>
      <SubjectId>1AAAAAAAAABBBBBBBBBCCCCCCCCCC</SubjectId>
      <ObjectType>HIT</ObjectType>
      <QuestionId>Question_1</QuestionId>
      <Key>PluralityAnswer</Key>
      <Value>true</Value>
    </ReviewResult>
    <ReviewResult>
      <SubjectId>1AAAAAAAAABBBBBBBBBCCCCCCCCCC</SubjectId>
      <ObjectType>HIT</ObjectType>
      <QuestionId>Question_1</QuestionId>
      <Key>PluralityLevel</Key>
      <Value>100</Value>
    </ReviewResult>
    <ReviewAction>
      <ActionName>approve</ActionName>
      <ObjectId>1DDDDDDDDDEEEEEEEEEEFFFFFFFFFF</ObjectId>
      <ObjectType>Assignment</ObjectType>
      <Status>SUCCEEDED</Status>
      <Result>Already approved</Result>
    </ReviewAction>
    <ReviewAction>
      <ActionName>approve</ActionName>
      <ObjectId>1GGGGGGGGGHHHHHHHHHIIIIIIIIII</ObjectId>
      <ObjectType>Assignment</ObjectType>
      <Status>FAILED</Status>
      <Result>Assignment was in an invalid state for this operation.</Result>
      <ErrorCode>AWS.MechanicalTurk.InvalidAssignmentState</ErrorCode>
    </ReviewAction>
  </HITReviewReport>
</GetReviewResultsForHITResult>
```

GrantBonus

Description

The `GrantBonus` operation issues a payment of money from your account to a Worker. This payment happens separately from the reward you pay to the Worker when you approve the Worker's assignment.

The `GrantBonus` operation requires the Worker's ID and the assignment ID as parameters to initiate payment of the bonus.

You must include a message that explains the reason for the bonus payment, as the Worker may not be expecting the payment.

Amazon Mechanical Turk collects a fee for bonus payments, similar to the HIT listing fee. This operation fails if your account does not have enough funds to pay for both the bonus and the fees.

For information about Amazon Mechanical Turk pricing and fee amounts, see Amazon Mechanical Turk Pricing.

Request Parameters

The `GrantBonus` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GrantBonus` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: GrantBonus Default: None	Yes
`WorkerId`	The ID of the Worker being paid the bonus, as returned in the assignment data of the GetAssignmentsForHIT (p. 52) operation. Type: String Default: None	Yes
`AssignmentId`	The ID of the assignment for which this bonus is paid, as returned in the assignment data of the GetAssignmentsForHIT (p. 52) operation. Type: String Default: None	Yes
`BonusAmount`	The bonus amount to pay. Type: Price (p. 157) data structure Default: None	Yes

Name	Description	Required
`Reason`	A message that explains the reason for the bonus payment. The Worker receiving the bonus can see this message. Type: String Default: None	Yes
`UniqueRequestToken`	A unique identifier for this request, which allows you to retry the call on error without granting multiple bonuses. This is useful in cases such as network timeouts where it is unclear whether or not the call succeeded on the server. If the bonus already exists in the system from a previous call using the same `UniqueRequestToken`, subsequent calls will return an error with a message containing the request ID. Type: String Default: None Constraints: must not be longer than 64 characters in length. Note: It is your responsibility to ensure uniqueness of the token. The unique token expires after 24 hours. Subsequent calls using the same `UniqueRequestToken` made after the 24 hour limit could grant multiple bonuses.	No

Response Elements

A successful request for the `GrantBonus` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`GrantBonusResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `GrantBonus` operation.

Sample Request

The following example of a call to the `GrantBonus` operation pays a bonus of $5 to a Worker.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GrantBonus
```

```
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&WorkerId=AZ3456EXAMPLE
&AssignmentId=123RVWYBAZW00EXAMPLE456RVWYBAZW00EXAMPLE
&BonusAmount.1.Amount=5
&BonusAmount.1.CurrencyCode=USD
&Reason=Thanks%20for%20doing%20great%20work!
```

Sample Response

The following is an example response.

```
<GrantBonusResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</GrantBonusResult>
```

GrantQualification

Description

The `GrantQualification` operation grants a Worker's request for a Qualification.

Only the owner of the Qualification type can grant a Qualification request for that type.

Request Parameters

`GrantQualification` accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `GrantQualification` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: GrantQualification Default: None	Yes
`QualificationRequestId`	The ID of the Qualification request, as returned by the GetQualificationRequests (p. 72) operation. Type: String Default: None	Yes
`IntegerValue`	The value of the Qualification Type: Integer Default: 1 Conditions: You can omit this value if you are using the presence or absence of the Qualification as the basis for a HIT requirement.	Conditional

Response Elements

A successful request for the `GrantQualification` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`GrantQualificationResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `GrantQualification` operation.

Sample Request

The following example grants a Qualification to a user.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=GrantQualification
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&QualificationRequestId=789RVWYBAZW00EXAMPLE951RVWYBAZW00EXAMPLE
&IntegerValue=95
```

Sample Response

The following is an example response.

```
<GrantQualificationResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</GrantQualificationResult>
```

NotifyWorkers

Description

The `NotifyWorkers` operation sends an email to one or more Workers that you specify with the Worker ID.

You can specify up to 100 Worker IDs to send the same message with a single call to the `NotifyWorkers` operation.

Note
The `NotifyWorkers` operation will send a notification email to a Worker only if you have previously approved or rejected work from the Worker.

Request Parameters

The `NotifyWorkers` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `NotifyWorkers` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: NotifyWorkers Default: None	Yes
`Subject`	The subject line of the email message to send. Type: String Default: None Constraints: can include up to 200 characters.	Yes
`MessageText`	The text of the email message to send Type: String Default: None Constraints: can include up to 4,096 characters	Yes
`WorkerId`	The ID of the Worker to notify, as returned by the GetAssignmentsForHIT (p. 52) operation. Type: String Default: None Constraints: You can repeat this parameter up to 100 times to notify multiple Workers.	Yes

Response Elements

A successful request for the `NotifyWorkers` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`NotifyWorkersResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `NotifyWorkers` operation.

Sample Request

The following example sends an email message to three Workers.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&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
```

Sample Response

The following is an example response.

```
<NotifyWorkersResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</NotifyWorkersResult>
```

RegisterHITType

Description

The `RegisterHITType` operation creates a new HIT type.

The `RegisterHITType` operation lets you be explicit about which HITs ought to be the same type. It also gives you error checking, to ensure that you call the CreateHIT (p. 21) operation with a valid HIT type ID.

If you register a HIT type with values that match an existing HIT type, the HIT type ID of the existing type will be returned.

Request Parameters

The `RegisterHITType` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `RegisterHITType` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: RegisterHITType Default: None	Yes
`Title`	The title for HITs of this type. A `Title` parameter should be short and descriptive about the kind of task the HIT contains. On the Amazon Mechanical Turk web site, the HIT title appears in search results, and everywhere the HIT is mentioned. Type: String Default: None Constraints: can be up to 128 characters in length	Yes
`Description`	A general description of HITs of this type A `Description` includes detailed information about the kind of task the HIT contains. On the Amazon Mechanical Turk web site, the HIT description appears in the expanded view of search results, and in the HIT and assignment screens. A good description gives the user enough information to evaluate the HIT before accepting it. It should not include instructions for completing the HIT.	Yes

Name	Description	Required
	Type: String Default: None Constraints: must be less than 2,000 characters in length	
`Reward`	The amount of money the Requester will pay a user for successfully completing a HIT of this type. Type: a Price (p. 157) data structure Default: None	Yes
`AssignmentDurationInSeconds`	The amount of time a Worker has to complete a HIT of this type after accepting it. Type: positive integer Default: None Constraints: any integer between 30 (30 seconds) and 3153600 (365 days)	Yes
`Keywords`	One or more words or phrases that describe a HIT of this type, separated by commas. Searches for words similar to the keywords are more likely to return the HIT in the search results. Type: String Default: None Constraints: The complete string, including commas and spaces, must be fewer than 1,000 characters.	No
`AutoApprovalDelayInSeconds`	An amount of time, in seconds, after an assignment for a HIT of this type has been submitted, that the assignment becomes **Approved** automatically, unless the Requester explicitly rejects it. Type: positive integer Default: 2592000 (30 days) Constraints: must be between 0 (immediate) and 2592000 (30 days).	No

Name	Description	Required
QualificationRequirement	A condition that a Worker's Qualifications must meet before the Worker is allowed to accept and complete a HIT of this type. Type: a QualificationRequirement (p. 163) data structure. Default: None Constraints: there can be no more than 10 QualificationRequirement data structures for each HIT.	No

Response Elements

A successful request for the RegisterHITType operation has a RegisterHITTypeResult element in the response.

The RegisterHITTypeResult element contains the following elements:

Name	Description
HITTypeId	The ID of the newly registered HIT type Type: String Default: None

Examples

The following example shows how to use the GetHITsForQualificationType operation.

Sample Request

The following example registers a new HIT type.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=RegisterHITType
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&Title=Location%20and%20Photograph%20Identification
&Description=Select%20the%20image%20that%20best%20represents...
&Reward.1.Amount=5
&Reward.1.CurrencyCode=USD
&AssignmentDurationInSeconds=30
&Keywords=location,%20photograph,%20image,%20identification,%20opinion
```

Sample Response

The following is an example response.

```
<RegisterHITTypeResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <HITTypeId>KZ3GKTRXBWGYX8WXBW60</HITTypeId>
</RegisterHITTypeResult>
```

RejectAssignment

Description

The `RejectAssignment` operation rejects the results of a completed assignment.

You can include an optional feedback message with the rejection, which the Worker can see in the **Status** section of the web site. When you include a feedback message with the rejection, it helps the Worker understand why the assignment was rejected, and can improve the quality of the results the Worker submits in the future.

Only the Requester who created the HIT can reject an assignment for the HIT.

Request Parameters

The `RejectAssignment` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `RejectAssignment` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: RejectAssignment Default: None	Yes
`AssignmentId`	The assignment ID Type: String Default: None	Yes
`RequesterFeedback`	A message for the Worker, which the Worker can see in the **Status** section of the web site. Type: String Default: None Constraints: can be up to 1024 characters, including multi-byte characters.	No

Response Elements

A successful request for the `RejectAssignment` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`RejectAssignmentResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `RejectAssignment` operation.

Sample Request

The following example rejects an assignment identified by its assignment ID.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=RejectAssignment
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&AssignmentId=123RVWYBAZW00EXAMPLE456RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<RejectAssignmentResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</RejectAssignmentResult>
```

RejectQualificationRequest

Description

The `RejectQualificationRequest` operation rejects a user's request for a Qualification.

You can provide a text message explaining why the request was rejected. The Worker who made the request can see this message.

Request Parameters

The `RejectQualificationRequest` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `RejectQualificationRequest` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: RejectQualificationRequest Default: None	Yes
`QualificationRequestId`	The ID of the Qualification request to reject, as returned from a call to the GetQualificationRequest (p. 72) operation. Type: String Default: None	Yes
`Reason`	A text message explaining why the request was rejected, to be shown to the Worker who made the request.	No

Response Elements

A successful request for the `RejectQualificationRequest` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`RejectQualificationRequestResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `RejectQualificationRequestType` operation.

Sample Request

The following example rejects a specified Qualification request.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=RejectQualificationRequest
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&QualificationRequestId=789RVWYBAZW00EXAMPLE951RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<RejectQualificationRequestResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</RejectQualificationRequestResult>
```

RevokeQualification

Description

The `RevokeQualification` operation revokes a previously granted Qualification from a user.

You can provide a text message explaining why the Qualification was revoked. The user who had the Qualification can see this message.

Request Parameters

The `RevokeQualification` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `RevokeQualification` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: RevokeQualification Default: None	Yes
`SubjectId`	The ID of the Worker who possesses the Qualification to be revoked. Type: String Default: None	Yes
`QualificationTypeId`	The ID of the Qualification type of the Qualification to be revoked. Type: String Default: None	Yes
`Reason`	A text message that explains why the Qualification was revoked. The user who had the Qualification sees this message. Type: String Default: None	No

Response Elements

A successful request for the `RevokeQualification` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
RevokeQualificationResult	Contains a Request element if the **Request** ResponseGroup is specified.

Examples

The following example shows how to use the RevokeQualification operation.

Sample Request

The following example revokes Qualification of the specified Qualification type for the specified user.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=RevokeQualification
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&SubjectId=AZ3456EXAMPLE
&QualificationTypeId=789RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<RevokeQualificationResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</RevokeQualificationResult>
```

SearchHITs

Description

The `SearchHITs` operation returns all of a Requester's HITs, on behalf of the Requester.

The operation returns HITs of any status, except for HITs that have been disposed of with the DisposeHIT (p. 38) operation or that have been auto-disposed.

> **Note**
> The `SearchHITs` operation does not accept any search parameters that filter the results. HITs are auto-disposed after 120 days and do not appear in `SearchHITs` responses after that time.

The operation sorts the results and divides them into numbered pages. The operation returns a single page of results. You can control sorting and pagination with parameters to the operation.

When (`PageNumber` x `PageSize`) is less than 100, you can get reliable results when you use any of the sort properties. If this number is greater than 100, use the **`Enumeration`** sort property for best results. The **`Enumeration`** sort property guarantees that all HITs are returned with no duplicates, but not in any specific order.

Request Parameters

The `SearchHITs` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `SearchHITs` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: SearchHITs Default: None	Yes
`SortProperty`	The field on which to sort the returned results Type: String Valid Values: Title | Reward | Expiration | CreationTime | Enumeration Default: CreationTime	No
`SortDirection`	The direction of the sort, used with the field specified by the `SortProperty` parameter. Type: String Valid Values: Ascending | Descending Default: Ascending	No

Name	Description	Required
PageSize	The number of HITs to include in a page of results. The complete sorted result set is divided into pages of this many HITs. Type: positive integer Valid Values: any integer between 1 and 100 Default: 10	No
PageNumber	The page of results to return. After the operation sorts the HITs and divides them into pages of size PageSize, the operation returns the page corresponding to PageNumber. Type: positive integer Default: 1	No

Response Elements

A successful request for the SearchHITs operation will have a SearchHITsResult element in the response.

The SearchHITsResult element contains the following elements:

Name	Description
NumResults	The number of HITs on this page in the filtered results list, equivalent to the number of HITs being returned by this call. Type: non-negative integer
PageNumber	The number of this page in the filtered results list. Type: positive integer
TotalNumResults	The total number of HITs in the filtered results list based on this call. Type: non-negative integer
HIT	The HIT. The response includes one HIT element for each HIT the query returns. Type: a HIT (p. 144) data structure.

Examples

The following example shows how to use the GetHITsForQualificationType operation.

Sample Request

The following example queries all of the HITs for a Requester. The example uses default values for sorting and pagination.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Operation=SearchHITs
&Signature=[signature for this request]
&Timestamp=[your system's local time]
```

Sample Response

The following is an example response.

```
<SearchHITsResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <NumResults>2</NumResults>
  <TotalNumResults>2</TotalNumResults>
  <PageNumber>1</PageNumber>

  <HIT>
    <HITId>GBHZVQX3EHXZ2AYDY2T0</HITId>
    <HITTypeId>NYVZTQ1QVKJZXCYZCZVZ</HITTypeId>
    <CreationTime>2009-04-22T00:17:32Z</CreationTime>
    <Title>Location</Title>
    <Description>Select the image that best represents</Description>
    <HITStatus>Reviewable</HITStatus>
    <MaxAssignments>1</MaxAssignments>
    <Reward>
      <Amount>5.00</Amount>
      <CurrencyCode>USD</CurrencyCode>
      <FormattedPrice>$5.00</FormattedPrice>
    </Reward>
    <AutoApprovalDelayInSeconds>2592000</AutoApprovalDelayInSeconds>
    <Expiration>2009-04-29T00:17:32Z</Expiration>
    <AssignmentDurationInSeconds>30</AssignmentDurationInSeconds>
    <NumberOfAssignmentsPending>0</NumberOfAssignmentsPending>
    <NumberOfAssignmentsAvailable>0</NumberOfAssignmentsAvailable>
    <NumberOfAssignmentsCompleted>1</NumberOfAssignmentsCompleted>
  </HIT>

  <HIT>
    <HITId>ZZRZPTY4ERDZWJ868JCZ</HITId>
    <HITTypeId>NYVZTQ1QVKJZXCYZCZVZ</HITTypeId>
    <CreationTime>2009-07-07T00:56:40Z</CreationTime>
    <Title>Location</Title>
    <Description>Select the image that best represents</Description>
    <HITStatus>Assignable</HITStatus>
    <MaxAssignments>1</MaxAssignments>
    <Reward>
      <Amount>5.00</Amount>
      <CurrencyCode>USD</CurrencyCode>
      <FormattedPrice>$5.00</FormattedPrice>
    </Reward>
    <AutoApprovalDelayInSeconds>2592000</AutoApprovalDelayInSeconds>
    <Expiration>2009-07-14T00:56:40Z</Expiration>
    <AssignmentDurationInSeconds>30</AssignmentDurationInSeconds>
    <NumberOfAssignmentsPending>0</NumberOfAssignmentsPending>
```

```
    <NumberOfAssignmentsAvailable>1</NumberOfAssignmentsAvailable>
    <NumberOfAssignmentsCompleted>0</NumberOfAssignmentsCompleted>
  </HIT>
</SearchHITsResult>
```

Related Operations

- GetAssignmentsForHIT (p. 52)

SearchQualificationTypes

Description

The `SearchQualificationTypes` operation searches for Qualification types using the specified search query, and returns a list of Qualification types.

The operation sorts the results, divides them into numbered pages, and returns a single page of results. You can control sorting and pagination with parameters to the operation.

Request Parameters

`SearchQualificationTypes` accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `SearchQualificationTypes` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: SearchQualificationTypes Default: None	Yes
`Query`	A text query against all of the searchable attributes of Qualification types. Type: String Default: None. If not specified, the complete set of all Qualification types is considered for the results.	No
`SortProperty`	The field on which to sort the results returned by the operation. Type: String Valid Values: Name Default: Name	No
`SortDirection`	The direction of the sort used with the field specified by `SortProperty`. Type: String Valid Values: Ascending | Descending Default: Descending	No
`PageSize`	The number of Qualification types to include in a page of results. The operation divides the complete sorted result set into pages of this many Qualification types. Type: positive integer	No

Name	Description	Required
	Valid Values: any integer between 1 and 100. Default: 10	
`PageNumber`	The page of results to return. After the operation filters, sorts, and divides the Qualification types into pages of size `PageSize`, it returns page corresponding to `PageNumber` as the results of the operation. Type: positive integer Default: 1	No
`MustBeRequestable`	Specifies that only Qualification types that a user can request through the Amazon Mechanical Turk web site, such as by taking a Qualification test, are returned as results of the search. Some Qualification types, such as those assigned automatically by the system, cannot be requested directly by users. If `false`, all Qualification types, including those managed by the system, are considered for the search. Type: Boolean Valid Values: true | false Default: None	Yes
`MustBeOwnedByCaller`	Specifies that only Qualification types that the Requester created are returned. If `false`, the operation returns all Qualification types.	No

Response Elements

A successful request for the `SearchQualificationTypes` operation has a `SearchQualificationTypesResult` element in the response.

The `SearchQualificationTypesResult` element contains the elements described in the following table:

Name	Description
`NumResults`	The number of Qualification types on this page in the filtered results list, equivalent to the number of types this operation returns. Type: non-negative integer
`PageNumber`	The number of this page in the filtered results list. Type: positive integer
`TotalNumResults`	The total number of Qualification types in the filtered results list based on this call. Type: non-negative integer

Name	Description
QualificationType	The Qualification type. The response includes one QualificationType element for each Qualification type the query returns. Type: a QualificationType (p. 171) data structure.

Examples

The following example shows how to use the SearchQualificationTypes operation.

Sample Request

The following example performs a simple text query for Qualification types.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=SearchQualificationTypes
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&Query=English
```

Sample Response

The following is an example response.

```
<SearchQualificationTypesResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <NumResults>10</NumResults>
  <TotalNumResults>5813</TotalNumResults>
  <QualificationType>
    <QualificationTypeId>WKAZMYZDCYCZP412TZEZ</QualificationTypeId>
    <CreationTime>2009-05-17T10:05:15Z</CreationTime>
    <Name> WebReviews Qualification Master Test</Name>
    <Description>
    This qualification will allow you to earn more on the WebReviews HITs.
    </Description>
    <Keywords>WebReviews, webreviews, web reviews</Keywords>
    <QualificationTypeStatus>Active</QualificationTypeStatus>
      <Test>
        <QuestionForm xmlns="http://mechanicalturk.amazonaws.com/
AWSMechanicalTurkDataSchemas/2005-10-01/QuestionForm.xsd">
          <Overview>
          <Title>WebReviews Survey</Title>
          <Text>
          After you have filled out this survey you will be assigned one or more
 qualifications...
          </Text>
        </Overview>
        <Question>
          <QuestionIdentifier>age</QuestionIdentifier>
          <DisplayName>What is your age?</DisplayName>
          <IsRequired>true</IsRequired>
          <QuestionContent>
```

```
            <Text>
            Please choose the age group you belong to.
            </Text>
          </QuestionContent>
          <AnswerSpecification>
      <SelectionAnswer>
        <StyleSuggestion>radiobutton</StyleSuggestion>
        <Selections>
          <Selection>
            <SelectionIdentifier>0018</SelectionIdentifier>
            <Text>-18</Text>
          </Selection>
        <Selection>
          <SelectionIdentifier>5160</SelectionIdentifier>
          <Text>51-60</Text>
        </Selection>
        <Selection>
          <SelectionIdentifier>6000</SelectionIdentifier>
          <Text>60+</Text>
        </Selection>
      </Selections>
       </SelectionAnswer>
         </AnswerSpecification>
      </Question>
    </QuestionForm>
    </Test>
    <TestDurationInSeconds>1200</TestDurationInSeconds>
  </QualificationType>
</SearchQualificationTypesResult>
```

SendTestEventNotification

Description

The `SendTestEventNotification` operation causes Amazon Mechanical Turk to send a notification message as if a HIT event occurred, according to the provided notification specification. This allows you to test notifications without setting up notifications for a real HIT type and trying to trigger them using the website.

When you call this operation, the service sends the test notification immediately.

Request Parameters

The `SendTestEventNotification` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `SendTestEventNotification` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: SendTestEventNotification Default: None	Yes
`Notification`	The notification specification to test. This value is identical to the value you would provide to the SetHITTypeNotification (p. 129) operation when you establish the notification specification for a HIT type. Type: a Notification (p. 175) data structure Default: None	Yes
`TestEventType`	The event to simulate to test the notification specification. This event is included in the test message even if the notification specification does not include the event type. The notification specification does not filter out the test event. Type: an `EventType` element of the Notification (p. 175) data structure. Default: None	Yes

Response Elements

A successful request for the `SendTestEventNotification` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
SendTestEventNotificationResult	Contains a Request element if the **Request** ResponseGroup is specified.

Examples

The following example shows how to use the SendTestEventNotification operation.

Sample Request

The following example sends a notification message for a simulated AssignmentSubmitted event to an email address.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Operation=SendTestEventNotification
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&Notification.1.Destination=janedoe@example.com
&Notification.1.Transport=Email
&Notification.1.Version=2006-05-05
&Notification.1.EventType=AssignmentSubmitted
&TestEventType=AssignmentSubmitted
```

Sample Response

The following is an example response.

```
<SendTestEventNotificationResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</SendTestEventNotificationResult>
```

SetHITAsReviewing

Description

The `SetHITAsReviewing` operation updates the status of a HIT. If the status is **`Reviewable`**, this operation updates the status to **`Reviewing`**, or reverts a **`Reviewing`** HIT back to the **`Reviewable`** status.

You can update only HITs with status **`Reviewable`** to status **`Reviewing`**. Similarly, you can revert only **`Reviewing`** HITs back to status **`Reviewable`**.

> **Note**
> The `SetHITAsReviewing` operation does *not* toggle the status value. The default behavior is to promote a HIT from **`Reviewable`** to **`Reviewing`**. To revert a **`Reviewing`** HIT back to **`Reviewable`**, specify the `Revert` parameter with a value of **`true`**.

Request Parameters

The `SetHITAsReviewing` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `SetHITAsReviewing` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String ValidValues: SetHITAsReviewing Default: None	Yes
`HITId`	The ID of the HIT whose status is to be updated. Type: String Default: None	Yes
`Revert`	Specifies whether to update the HIT `Status` from **`Reviewing`** to **`Reviewable`**. Type: Boolean Valid Values: true | false Default: false; the operation promotes the HIT from **`Reviewable`** to **`Reviewing`**.	No

Response Elements

A successful request for the `SetHITAsReviewing` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
SetHITAsReviewingResult	Contains a Request element if the **Request** ResponseGroup is specified.

Examples

The following example shows how to use the SetHITAsReviewingResult operation.

Sample Request

The following example updates a HIT with **Reviewable** status to **Reviewing** status.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=SetHITAsReviewing
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITId=123RVWYBAZW00EXAMPLE
```

Sample Response

The following is an example response.

```
<SetHITAsReviewingResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</SetHITAsReviewingResult>
```

SetHITTypeNotification

Description

The `SetHITTypeNotification` operation creates, updates, disables or re-enables notifications for a HIT type.

If you call the `SetHITTypeNotification` operation for a HIT type that already has a notification specification, the operation replaces the old specification with a new one.

You can call the `SetHITTypeNotification` operation to enable or disable notifications for the HIT type, without having to modify the notification specification itself.

You can call this operation at any time to change the value of the of the `Active` parameter of a HIT type. You can specify changes to the `Active` status without specifying a new notification specification (the `Notification` parameter).

To change the `Active` status of a HIT type's notifications, the HIT type must already have a notification specification, or one must be provided in the same call to `SetHITTypeNotification`.

Note
After you make the call to `SetHITTypeNotification`, it can take up to five minutes for changes to a HIT type's notification specification to take effect.

Request Parameters

The `SetHITTypeNotification` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `SetHITTypeNotification` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: SetHITTypeNotification Default: None	Yes
`HITTypeId`	The ID of the HIT type whose notification specification is being updated, as returned by the RegisterHITType (p. 107) operation. Type: String Default: None	Yes
`Notification`	The notification specification for the HIT type. Type: a Notification (p. 175) data structure. Default: None. Constraint: You must specify either the `Notification` parameter or the `Active` parameter for the call to `SetHITTypeNotification` to succeed.	Yes

Name	Description	Required
`Active`	Specifies whether notifications are sent for HITs of this HIT type, according to the notification specification. Type: Boolean Valid Values: true | false Default: None. If omitted, the active status of the HIT type's notification specification is unchanged Constraint: You must specify either the `Notification` parameter or the `Active` parameter for the call to `SetHITTypeNotification` to succeed.	No

Response Elements

A successful request for the `SetHITTypeNotification` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`SetHITTypeNotificationResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `SetHITTypeNotification` operation.

Sample Request

The following example sets the notification specification for a HIT type to send the Requester an email whenever a Worker submits an assignment for a HIT of the specified type.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Operation=SetHITTypeNotification
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&HITTypeId=T100CN9P324W00EXAMPLE
&Notification.1.Destination=janedoe@example.com
&Notification.1.Transport=Email
&Notification.1.Version=2006-05-05
&Notification.1.EventType=AssignmentSubmitted
```

Sample Response

The following is an example response.

```
<SetHITTypeNotificationResult>
  <Request>
    <IsValid>True</IsValid>
```

```
  </Request>
</SetHITTypeNotificationResult>
```

UnblockWorker

Description

The `UnblockWorker` operation allows you to reinstate a blocked Worker to work on your HITs. This operation reverses the effects of the BlockWorker (p. 16) operation.

You need the Worker ID to use this operation. If the Worker ID is missing or invalid, this operation fails and returns the message "WorkerId is invalid." If the specified Worker is not blocked, this operation returns successfully.

Request Parameters

The `UnblockWorker` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `UnblockWorker` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: UnblockWorker Default: None	Yes
`WorkerId`	The ID of the Worker to unblock. Type: String Default: None	Yes
`Reason`	A message that explains the reason for unblocking the Worker. The Worker does not see this message. Type: String Default: None	No

Response Elements

A successful request for the `UnblockWorker` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`UnblockWorkerResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `UnblockWorker` operation.

Sample Request

The following example unblocks a Worker and allows that Worker to work on your HITs.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=UnblockWorker
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&WorkerId=AZ3456EXAMPLE
```

Sample Response

The following is an example response.

```
<UnblockWorkerResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</UnblockWorkerResult>
```

UpdateQualificationScore

Description

The `UpdateQualificationScore` operation changes the value of a Qualification previously granted to a Worker.

Only the owner of a Qualification type can update the score of a Qualification of that type.

The Worker must have already been granted a Qualification of the given Qualification type before the score can be updated.

Request Parameters

The `UpdateQualificationScore` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

The following parameters are specific to the `UpdateQualificationScore` operation:

Name	Description	Required
`Operation`	The name of the operation Type: String Valid Values: UpdateQualificationScore Default: None	Yes
`QualificationTypeId`	The ID of the Qualification type, as returned by CreateQualificationType (p. 32) operation. Type: String Default: None	Yes
`SubjectId`	The ID of the Worker whose Qualification is being updated, as returned by the GetAssignmentsForHIT (p. 52) operation. Type: String Default: None	Yes
`IntegerValue`	The new value for the Qualification. Type: Integer Default: None	Yes

Response Elements

A successful request for the `UpdateQualificationScore` operation returns with no errors. The response includes the elements described in the following table. The operation returns no other data.

Name	Description
`UpdateQualificationScoreResult`	Contains a `Request` element if the **`Request`** `ResponseGroup` is specified.

Examples

The following example shows how to use the `UpdateQualificationScore` operation.

Sample Request

The following example changes the value of a Qualification of the specified type for the specified Worker.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=UpdateQualificationScore
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&QualificationTypeId=789RVWYBAZW00EXAMPLE
&SubjectId=AZ3456EXAMPLE
&IntegerValue=70
```

Sample Response

The following is an example response.

```
<UpdateQualificationScoreResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
</UpdateQualificationScoreResult>
```

UpdateQualificationType

Description

The `UpdateQualificationType` operation modifies the attributes of an existing Qualification type, which is represented by a QualificationType (p. 171) data structure. Only the owner of a Qualification type can modify its attributes.

Most attributes of a Qualification type can be changed after the type has been created. However, the `Name` and `Keywords` fields cannot be modified. The `RetryDelayInSeconds` parameter can be modified or added to change the delay or to enable retries, but `RetryDelayInSeconds` cannot be used to disable retries.

Each Qualification type name must be unique across all of your Qualification types. If you want to reuse the name of an existing Qualification type, you must first dispose of the existing Qualification type using DisposeQualificationType (p. 40) and then create a new Qualification type with the name of the disposed type using CreateQualificationType (p. 32).

If you want to disable retries, you must dispose of the existing retry-enabled Qualification type using DisposeQualificationType (p. 40) and then create a new Qualification type with retries disabled using CreateQualificationType (p. 32).

You can use this operation to update the test for a Qualification type. The test is updated based on the values specified for the `Test`, `TestDurationInSeconds` and `AnswerKey` parameters. All three parameters specify the updated test. If you are updating the test for a type, you must specify the `Test` and `TestDurationInSeconds` parameters. The `AnswerKey` parameter is optional; omitting it specifies that the updated test does not have an answer key.

If you omit the `Test` parameter, the test for the Qualification type is unchanged. There is no way to remove a test from a Qualification type that has one. If the type already has a test, you cannot update it to be `AutoGranted`. If the Qualification type does not have a test and one is provided by an update, the type will henceforth have a test.

If you want to update the test duration or answer key for an existing test without changing the questions, you must specify a `Test` parameter with the original questions, along with the updated values.

If you provide an `AnswerKey` parameter, Amazon Mechanical Turk processes requests for the Qualification automatically, and assigns the Worker a Qualification with a value calculated from the `AnswerKey` and the answers submitted by the Worker.

If you provide an updated `Test` but no `AnswerKey`, the new test will not have an answer key. Requests for such Qualifications must be granted manually.

You can also update the `AutoGranted` and `AutoGrantedValue` attributes of the Qualification type.

Note
A Qualification type cannot be configured with both the `Test` parameter specified and the `AutoGranted` parameter set to **`true`**. Currently, there is no way to remove a test from a Qualification type that has one. A Qualification type with a test cannot be re-configured to use the auto-granting feature.

Request Parameters

The `UpdateQualificationType` operation accepts parameters common to all operations. Some common parameters are required. See Common Parameters (p. 6) for more information.

Note
If an optional request parameter is not specified, the `UpdateQualificationType` operation does not modify the corresponding field of the QualificationType (p. 171) being updated.

The following parameters are specific to the `UpdateQualificationType` operation:

Name	Description	Required
`Operation`	The name of the operation. Type: String Valid Values: UpdateQualificationType Default: None	Yes
`QualificationTypeId`	The ID of the Qualification type to update. Type: String Default: None	Yes
`RetryDelayInSeconds`	The amount of time, in seconds, that Workers must wait after requesting a Qualification of the specified Qualification type before they can retry the Qualification request. It is not possible to disable retries for a Qualification type after it has been created with retries enabled. If you want to disable retries, you must dispose of the existing retry-enabled Qualification type using DisposeQualificationType (p. 40) and then create a new Qualification type with retries disabled using CreateQualificationType (p. 32). Type: Non-negative integer Default: None. If not specified, the `RetryDelayInSeconds` value of the specified Qualification type will not be changed.	No
`QualificationTypeStatus`	The new status of the Qualification type. Type: String Valid Values: Active | Inactive Default: None	No
`Description`	The new description of the Qualification type. Type: String Default: None	No
`Test`	The questions for a Qualification test a Worker must answer correctly to obtain a Qualification of this type. Type: A QuestionForm (p. 191) data structure. Default: None	No

Name	Description	Required
	Constraints: Must not be longer than 65535 bytes. Cannot be specified if `AutoGranted` is **`true`**. If you update the `AnswerKey`, you must provide the `Test` parameter, even if the test has not changed. If you update the `TestDurationInSeconds`, you must provide the `Test` parameter, even if the test has not changed.	
`AnswerKey`	The answers to the Qualification test specified in the `Test` parameter. Type: An AnswerKey (p. 212) data structure. Default: None Constraints: Must not be longer than 65535 bytes.	No
`TestDurationInSeconds`	The amount of time, in seconds, that a Worker has to complete the Qualification test, starting from when the Worker requested the Qualification. Type: Positive integer Default: None Conditions: Required if the `Test` parameter is specified.	Conditional
`AutoGranted`	Specifies whether requests for the Qualification type will be granted immediately, without prompting the Worker with a Qualification test. Type: Boolean Valid Values: true | false Default: None Conditions: Cannot be **`true`** if `Test` is specified.	Conditional
`AutoGrantedValue`	The Qualification value to use if `AutoGranted` is **`true`**. Type: Integer Default: 1	No

Response Elements

A successful request for the `UpdateQualificationType` operation has a `QualificationType` element in the response, as described in the following table:

Name	Description
`QualificationType`	Contains a QualificationType (p. 171) data structure.

Name	Description
	Type: A QualificationType (p. 171) data structure.

Examples

The following example shows how to use the `UpdateQualificationType` operation.

Sample Request

The following example changes the `QualificationTypeStatus` of a Qualification type.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
&AWSAccessKeyId=[the Requester's Access Key ID]
&Version=2017-01-17
&Operation=UpdateQualificationType
&Signature=[signature for this request]
&Timestamp=[your system's local time]
&QualificationTypeId=789RVWYBAZW00EXAMPLE
&QualificationTypeStatus=Inactive
```

Sample Response

The following is an example response.

```
<UpdateQualificationTypeResult>
  <Request>
    <IsValid>True</IsValid>
  </Request>
  <QualificationType>
    <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
    <CreationTime>2009-06-15T12:00:01Z</CreationTime>
    <Name>EnglishWritingAbility</Name>
    <Description>The ability to write and edit text...</Description>
    <Keywords>English, text, write, edit, language</Keywords>
    <QualificationTypeStatus>Active</QualificationTypeStatus>
    <RetryDelayInSeconds>86400</RetryDelayInSeconds>
    <IsRequestable>true</IsRequestable>
  </QualificationType>
</UpdateQualificationTypeResult>
```

Related Operations

- AssignQualification (p. 14)
- CreateQualificationType (p. 32)
- DisposeQualificationType (p. 40)
- GetQualificationType (p. 77)

Data Structures

Topics

The Amazon Mechanical Turk API uses several common data structures in its operation request and response structures. For easy reference, these structures are documented in separate articles. For more information, refer to their corresponding operations.

Assignment

Description

The Assignment data structure represents a single assignment of a HIT to a Worker. The assignment tracks the Worker's efforts to complete the HIT, and contains the results for later retrieval.

The Assignment data structure is used as a response element for the following operations:

- GetAssignment (p. 49)
- GetAssignmentsForHIT (p. 52)

Elements

The Assignment structure can contain the following elements.

Name	Description	Required
`AssignmentId`	A unique identifier for the assignment Type: String Default: None	No
`WorkerId`	The ID of the Worker who accepted the HIT. Type: String	No

Name	Description	Required
	Default: None	
`HITId`	The ID of the HIT Type: String Default: None	No
`AssignmentStatus`	The status of the assignment Type: String Valid Values: Submitted | Approved | Rejected Default: None	No
`AutoApprovalTime`	If results have been submitted, `AutoApprovalTime` is the date and time the results of the assignment results are considered **Approved** automatically if they have not already been explicitly approved or rejected by the Requester. This value is derived from the auto-approval delay specified by the Requester in the HIT. This value is omitted from the assignment if the Worker has not yet submitted results. Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **`2005-01-31T23:59:59Z`**. Default: None	No
`AcceptTime`	The date and time the Worker accepted the assignment. Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **`2005-01-31T23:59:59Z`**. Default: None	No
`SubmitTime`	If the Worker has submitted results, `SubmitTime` is the date and time the assignment was submitted. This value is omitted from the assignment if the Worker has not yet submitted results. Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **`2005-01-31T23:59:59Z`**. Default: None	No

Name	Description	Required
`ApprovalTime`	If the Worker has submitted results and the Requester has approved the results, `ApprovalTime` is the date and time the Requester approved the results. This value is omitted from the assignment if the Requester has not yet approved the results. Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **`2005-01-31T23:59:59Z`**. Default: None	No
`RejectionTime`	If the Worker has submitted results and the Requester has rejected the results, `RejectionTime` is the date and time the Requester rejected the results. Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **`2005-01-31T23:59:59Z`**. Default: None. This value is omitted from the assignment if the Requester has not yet rejected the results.	No
`Deadline`	The date and time of the deadline for the assignment. This value is derived from the deadline specification for the HIT and the date and time the Worker accepted the HIT. Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **`2005-01-31T23:59:59Z`**. Default: None	No
`Answer`	The Worker's answers submitted for the HIT contained in a `QuestionFormAnswers` document, if the Worker provides an answer. If the Worker does not provide any answers, `Answer` may contain a `QuestionFormAnswers` document, or `Answer` may be empty. Type: a QuestionFormAnswers (p. 211) data structure Default: None	No
`RequesterFeedback`	The feedback string included with the call to the ApproveAssignment (p. 11) operation or the RejectAssignment (p. 111) operation, if the Requester approved or rejected the assignment and specified feedback. Type: String Default: None. This field is not returned with assignment data by default. To request this field, specify a response group of **`AssignmentFeedback`**. For information about response groups, see Common Parameters (p. 6).	No

Example

The following example shows an Assignment data structure returned by the GetAssignmentsForHIT (p. 52) operation. The GetAssignmentsForHIT (p. 52) operation returns zero or more `Assignment` elements for a **Reviewable** HIT.

```
<Assignment>
  <AssignmentId>123RVWYBAZW00EXAMPLE456RVWYBAZW00EXAMPLE</AssignmentId>
  <WorkerId>AZ3456EXAMPLE</WorkerId>
  <HITId>123RVWYBAZW00EXAMPLE</HITId>
  <AssignmentStatus>Submitted</AssignmentStatus>
  <Deadline>2005-12-01T23:59:59Z</Deadline>
  <AcceptTime>2005-12-01T12:00:00Z</AcceptTime>
  <SubmitTime>2005-12-07T23:59:59Z</SubmitTime>
  <Answer>
    &lt;QuestionFormAnswers&gt;
      [XML-encoded Answer data]
    &lt;/QuestionFormAnswers&gt;
  </Answer>
</Assignment>
```

HIT

Description

The HIT data structure represents a single HIT, including all the information necessary for a Worker to accept and complete the HIT.

The HIT data structure is used as a response element for the following operations:

- CreateHIT (p. 21)
- DisableHIT (p. 36)
- GetHIT (p. 64)
- GetReviewableHITs (p. 89)
- SearchHITs (p. 117)

HITs and Response Groups

Operations that return a HIT data structure use response groups to determine how much information to return. As described in Common Parameters (p. 6), the `ResponseGroup` parameter specifies which sets of elements the service should return, as a set of named groups. For example, the `Request` response group includes the contents of the operation request in the response.

For the HIT data structure, the `Minimal` response group returns the `HITId`. For information about the contents of other HIT data structure related response group content, see Common Parameters (p. 6).

The `GetHIT` operation returns the **`HITDetail`**, **`HITQuestion`** and **`Minimal`** response groups by default. The **`HITAssignmentSummary`** response group is off by default.

The `SearchHITs` operation includes **`HITDetail`**, **`Minimal`**, and **`HITAssignmentSummary`** as default response groups. You can also specify **`HITQuestion`** with `SearchHITs`.

`CreateHIT` and `DisableHIT` can also return additional HIT fields, but their default is **`Minimal`**.

Currently, the `GetReviewableHITs` operation only supports the `Minimal` response group. To retrieve additional HIT data for HITs returned by this operation, use the HIT IDs in the results with `GetHIT`.

Elements

The HIT structure can contain the elements described in the following table.

Name	Description	Required
`HITId`	A unique identifier for the HIT. The CreateHIT (p. 21) operation gives a HIT the HIT ID and the HIT retains that ID forever. Type: String Default: None	No
`HITTypeId`	The ID of the HIT type of this HIT Type: String Default: None	No

Name	Description	Required
HITGroupId	The ID of the HIT Group of this HIT Type: String Default: None	No
HITLayoutId	The ID of the HIT Layout of this HIT Type: String Default: None	No
CreationTime	The date and time the HIT was created Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as 2012-01-31T23:59:59Z. Default: None	No
Title	The title of the HIT Type: String Default: None	No
Description	A general description of the HIT Type: String Default: None	No
Keywords	One or more words or phrases that describe the HIT, separated by commas. Search terms similar to the keywords of a HIT are more likely to have the HIT in the search results. Type: String Default: None	No
HITStatus	The status of the HIT and its assignments Type: String Valid Values: Assignable | Unassignable | Reviewable | Reviewing | Disposed Default: None	No
Reward	The amount of money the Requester will pay a Worker for successfully completing the HIT. Type: a Price (p. 157) data structure Default: None	No

Name	Description	Required
`LifetimeInSeconds`	The amount of time, in seconds, after which the HIT is no longer available for users to accept. The HIT becomes unavailable even if the requested number of assignments, specified by `MaxAssignments`, has not been completed. Type: positive integer Default: None	No
`AssignmentDurationInSeconds`	The length of time, in seconds, that a Worker has to complete the HIT after accepting it. Type: positive integer Default: None	No
`MaxAssignments`	The number of times the HIT can be accepted and completed before the HIT becomes unavailable. Type: positive integer Default: 1	No
`AutoApprovalDelayInSeconds`	The amount of time, in seconds, after the Worker submits an assignment for the HIT that the results are automatically approved by Amazon Mechanical Turk. This is the amount of time the Requester has to reject an assignment submitted by a Worker before the assignment is auto-approved and the Worker is paid. Type: positive integer Default: None	No
`Expiration`	The date and time the HIT expires Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as `2012-01-31T23:59:59Z`. Default: None	No

Name	Description	Required
`QualificationRequirement`	A condition that a Worker's Qualifications must meet in order to accept the HIT. A HIT can have between zero and ten Qualification requirements. All requirements must be met by a Worker's Qualifications for the Worker to accept the HIT. Type: a QualificationRequirement (p. 163) data structure. Default: None	No
`Question`	The data the Worker completing the HIT uses produce the results. Type: either a QuestionForm (p. 191) or an ExternalQuestion (p. 216) data structure. Default: None	No
`RequesterAnnotation`	An arbitrary data field the Requester who created the HIT can use. This field is visible only to the creator of the HIT. Type: String Default: None	No
`HITReviewStatus`	Indicates the review status of the HIT. Type: String Valid Values: NotReviewed | MarkedForReview | ReviewedAppropriate | ReviewedInappropriate Default: None	No
`NumberofAssignmentsPending`	The number of assignments for this HIT that are being previewed or have been accepted by Workers, but have not yet been submitted, returned, or abandoned. Type: non-negative integer Default: None Conditions: This element is returned only if the `HITAssignmentSummary` response group is specified.	Conditional

Name	Description	Required
NumberofAssignmentsAvailable	The number of assignments for this HIT that are available for Workers to accept Type: non-negative integer Default: None Conditions: This element is returned only if the HITAssignmentSummary response group is specified.	Conditional
NumberofAssignmentsCompleted	The number of assignments for this HIT that have been approved or rejected. Type: non-negative integer Default: None Conditions: This element is returned only if the HITAssignmentSummary response group is specified.	Conditional

Example

The following example shows a HIT data structure returned by the CreateHIT (p. 21) operation. The CreateHIT (p. 21) operation returns an element named HIT that represents the HIT that was created by the call.

```
<HIT>
  <HITId>123RVWYBAZW00EXAMPLE</HITId>
  <HITTypeId>T100CN9P324W00EXAMPLE</HITTypeId>
  <CreationTime>2005-06-30T23:59:59</CreationTime>
  <HITStatus>Assignable</HITStatus>
  <MaxAssignments>5</MaxAssignments>
  <AutoApprovalDelayInSeconds>86400</AutoApprovalDelayInSeconds>
  <LifetimeInSeconds>86400</LifetimeInSeconds>
  <AssignmentDurationInSeconds>300</AssignmentDurationInSeconds>
  <Reward>
    <Amount>25</Amount>
    <CurrencyCode>USD</CurrencyCode>
    <FormattedPrice>$0.25</FormattedPrice>
  </Reward>
  <Title>Location and Photograph Identification</Title>
  <Description>Select the image that best represents...</Description>
  <Keywords>location, photograph, image, identification, opinion</Keywords>
  <Question>
    &lt;QuestionForm&gt;
      [XML-encoded Question data]
    &lt;/QuestionForm&gt;
  </Question>
  <QualificationRequirement>
    <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
    <Comparator>GreaterThan</Comparator>
    <Value>18</Value>
  </QualificationRequirement>
  <HITReviewStatus>NotReviewed</HITReviewStatus>
</HIT>
```

HITLayoutParameter

Description

The `HITLayoutParameter` data structure defines parameter values used with a HITLayout. A HITLayout is a reusable Amazon Mechanical Turk project template used to provide Human Intelligence Task (HIT) question data for `CreateHIT`. For more information, see HITLayout (p. 223).

The `HITLayoutParameter` data structure is used in the results of the following operation:

- CreateHIT (p. 21)

Tip
The HITLayout (p. 223) is used to create an HTMLQuestion document. HITLayoutParameter values with reserved characters or invalid HTML markup may result in an invalid HTMLQuestion document.

Elements

The `HITLayout` data structure has a root element of `HITLayoutParameter`.

The `HITLayoutParameter` element contains the following elements:

Name	Description	Required
`Name`	The name of the parameter in the HITLayout. Type: String Default: None Constraints: Parameter names supplied in a HITLayoutParameter structure must be used in the referenced HITLayout.	Yes
`Value`	The value substituted for the parameter referenced in the HITLayout. Type: String Default: None	Yes

HIT Review Policy

Topics

Description

HIT Review Policy data structures represent HIT review policies, which you specify when you create a Human Intelligence Task (HIT). For more information about Review Policies, see Review Policies (p. 178).

The following two types of HIT Review Policy data structures are used when calling the CreateHIT (p. 21) operation.

- `AssignmentReviewPolicy` data structures set the review results and actions at the Assignment level. For more information, see Assignment Review Policies (p. 179).
- `HITReviewPolicy` data structures set the review results and actions at the HIT-level. For more information, see HIT Review Policies (p. 181).

The HIT Review Policy data structure is used in the following operation:

- `CreateHIT`

HIT Review Policy Elements

The HIT Review Policy data structures can contain the following elements.

Name	Description	Required
`PolicyName`	Name of a Review Policy. For policy names and descriptions, see Assignment Review Policies (p. 179) and HIT Review Policies (p. 181). Type: String Constraints: SimplePlurality/2011-09-01 or ScoreMyKnownAnswers/2011-09-01	Yes
`Parameter`	Name of the parameter from the Review policy. Type: Parameter data structure, for a description see the next section Parameter Elements.	

Parameter Elements

The Parameter data structure contains the elements described in the following table.

Name	Description	Required
`Key`	Name of the parameter from the list of Review Polices. For a list of valid parameter names, see Assignment Review Policies (p. 179) and HIT Review Policies (p. 181). Type: String Constraints: none Default: none	
`Value`	Value of the parameter. Type: Varies.	
`MapEntry`	This data structure is the data type for the `AnswerKey` parameter of the ScoreMyKnownAnswers/2011-09-01 Review Policy, see Assignment Review Policies (p. 179). Type: MapEntry data structure, for a description, see the next section MapEntry Elements.	

MapEntry Elements

The MapEntry element contains the elements described in the following table.

Name	Description	Required
`Key`	The QuestionID from the HIT that is used to identify which question requires Mechanical Turk to score as part of the ScoreMyKnownAnswers/2011-09-01 Review Policy. Type: String none	
`Value`	The answer to the question specified in the MapEntry `Key` element. Note: You can provide multiple `Value` elements, but the Worker must match all values in order for the answer to be scored correctly. Type: String	

Examples

The following example request shows the structure of a `CreateHIT` request using HIT Review Policy data structures. The example request is followed by an example response.

Sample Request

The following is an example request.

```
<CreateHITRequest>
```

```
    <HITTypeId>T100CN9P324W00EXAMPLE</HITTypeId>
    <Question>URL-encoded question data</Question>
    <LifetimeInSeconds>604800</LifetimeInSeconds>
    <AssignmentReviewPolicy>
        <PolicyName>ScoreMyKnownAnswers/2011-09-01</PolicyName>
        <Parameter>
            <Key>AnswerKey</Key>
            <MapEntry>
                <Key>QuestionId1</Key>   <!—correct answer is “B” -->
                <Value>B</Value>
            </MapEntry
            <MapEntry>
                <Key>QuestionId2</Key>  <!—correct answer is “A” -->
                <Value>A</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId1</Key>   <!—correct answer is “F” -->
                <Value>F</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId1</Key>   <!—correct answer is “C” -->
                <Value>C</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId1</Key>   <!—correct answer is “A” -->
                <Value>A</Value>
            </MapEntry
        <Parameter>
           <Key>ApproveIfKnownAnswerScoreIsAtLeast</Key>
           <Value>79</Value>
        </Parameter>
        <Parameter>
           <Key>ExtendIfKnownAnswerScoreIsLessThan</Key>
           <Value>79</Value>
        </Parameter>
        <Parameter>
           <Key>ExtendMaximumAssignments</Key>
           <Value>3</Value>
        </Parameter>
    </AssignmentReviewPolicy>
    <HITReviewPolicy>
        <PolicyName>SimplePlurality/2011-09-01</PolicyName>
        <Parameter>
           <Key>QuestionIDs</Key>
           <Value>questionid1</Value>
           <Value>questionid2</Value>
           <Value>questionid3</Value>
           <Value>questionid4</Value>
           <Value>questionid5</Value>
                 ..... <! Add your additional 10 questionIDs for a total of 15 questions>
        </Parameter>
        <Parameter>
           <Key>QuestionAgreementThreshold</Key>
           <Value>100</Value>
        </Parameter>
    </HITReviewPolicy>
</CreateHITRequest>
```

Sample Response

The following is an example response.

```
<CreateHITResponse>
  <OperationRequest>
```

```
    <RequestId>ece2785b-6292-4b12-a60e-4c34847a7916</RequestId>
  </OperationRequest>
  <HIT>
    <Request>
      <IsValid>True</IsValid>
    </Request>
    <HITId>GBHZVQX3EHXZ2AYDY2T0</HITId>
    <HITTypeId>NYVZTQ1QVKJZXCYZCZVZ</HITTypeId>
  </HIT>
</CreateHITResponse>
```

Locale

Description

The Locale data structure represents a geographical region or location.

The Locale data structure is used as part of the QualificationRequirement (p. 163) data structure when you specify a requirement based on the locale Qualification, and as part of the Qualification (p. 159) data structure that describes the value of a locale Qualification.

When used in a QualificationRequirement, the Locale data structure only needs to contain as much of the locale as the Worker needs to match to meet the requirement. For example, a requirement that the Worker be living anywhere in the United States would have only the `Country` field.

Note
Currently, a Locale data structure only supports the `Country` field and `Subdivision` field. Please note that subdivisions or states are only available for the United States of America.

Elements

The Locale structure can contain the elements described in the following table. When the structure is used in a request, elements described as **Required** must be included for the request to succeed.

Name	Description	Required
`Country`	The country of the locale. Type: A valid ISO 3166 country code. For example, the code **`US`** refers to the United States of America. Default: none	Yes
`Subdivision`	The state or subdivision of the locale. Type: Type: A valid ISO 3166-2 subdivision code. For example, the code CA refers to the state of California Default: none	Yes

Example

The following code sample indicates a locale in the United States.

```
<LocaleValue>
  <Country>US</Country>
</LocaleValue>
```

Example

The following code sample indicates a locale in the state of California in the United States of America.

```
<LocaleValue>
```

```
  <Country>US</Country>
  <Subdivision>CA</Subdivision>
</LocaleValue>
```

Price

Description

The Price data structure represents an amount of money in a given currency.

The Price data structure is used in the HIT data structure (p. 144).

The Price data structure is used as a parameter for the following operations:

- `CreateHIT`

When you call the CreateHIT (p. 21) operation, you must specify the `Amount` and `CurrencyCode` elements. The `FormattedPrice` element is only used in responses sent by the service.

Elements

The Price structure can contain the elements described in the following table:

Name	Description	Required
`Amount`	The amount of money, as a number. The amount is in the currency specified by the `CurrencyCode`. For example, if `CurrencyCode` is **`USD`**, the amount will be in United States dollars (e.g. **`12.75`** is $12.75 US). Type: Number Default: None	No
`CurrencyCode`	A code that represents the country and units of the currency. Its value is Type an ISO 4217 currency code, such as **`USD`** for United States dollars. Default: None Constraints: Currently, only **`USD`** is supported.	No
`FormattedPrice`	A textual representation of the price, using symbols and formatting appropriate for the currency. Symbols are represented using the Unicode character set. You do not need to specify `FormattedPrice` in a request. It is only provided by the service in responses, as a convenience to your application. Type: String Default: None	No

Example

The following example shows how you can pass a Price data structure in a call to the CreateHIT (p. 21) operation. The CreateHIT (p. 21) operation accepts parameters that describe the HIT being created,

including the reward the Worker will be paid for completing the HIT successfully. For CreateHIT (p. 21), the parameter name is **`Reward`**, and the value is a Price data structure.

In a SOAP request, the Price data structure is specified as the `Reward` parameter in XML:

```
<Reward>
  <Amount>0.32</Amount>
  <CurrencyCode>USD</CurrencyCode>
</Reward>
```

In a REST request, the components of the Price data structure are specified as separate parameters:

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
[...]
&Reward.1.Amount=0.32
&Reward.1.CurrencyCode=USD
```

Qualification

Description

The Qualification data structure represents a Qualification assigned to a user, including the Qualification type and the value (score).

The Qualification data structure is used as a response element for the following operations:

- GetQualificationScore (p. 75)
- GetQualificationsForQualificationType (p. 69)

Elements

The Qualification structure can contain the elements described in the following table. When the structure is used in a request, elements described as **Required** must be included for the request to succeed.

Name	Description	Required
`QualificationTypeId`	The ID of the Qualification type for the Qualification Type: String Default: None	Yes
`SubjectId`	The ID of the Worker who possesses the Qualification. Type: String Default: None	Yes
`GrantTime`	The date and time the Qualification was granted to the Worker. If the Worker's Qualification was revoked, and then re-granted based on a new Qualification request, `GrantTime` is the date and time of the last call to the GrantQualification (p. 103) operation. Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **`2005-01-31T23:59:59Z`** Default: None	Yes
`IntegerValue`	The value (score) of the Qualification, if the Qualification has an integer value. Type: Integer Default: None	No
`LocaleValue`	The value of the Qualification if the Qualification describes a geographical region or location. Type: a Locale (p. 155) data structure.	No

Name	Description	Required
	Default: None	
`Status`	The status of the Qualification Type: String Valid Values: Granted | Revoked Default: None	Yes

Example

The following example illustrates a Qualification with an integer value.

```
<Qualification>
  <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
  <SubjectId>AZ3456EXAMPLE</SubjectId>
  <GrantTime>2005-01-31T23:59:59Z</GrantTime>
  <IntegerValue>95</IntegerValue>
</Qualification>
```

QualificationRequest

Description

The QualificationRequest data structure represents a request a Worker has made for a Qualification.

The QualificationRequest data structure is used as a response element for the following operations:

- GetQualificationRequests (p. 72)

Elements

The QualificationRequest structure can contain the elements described in the following table:

Name	Description	Required
`QualificationRequestId`	The ID of the Qualification request, a unique identifier generated when the request was submitted. Type: String Default: None	No
`QualificationTypeId`	The ID of the Qualification type the Worker is requesting, as returned by the CreateQualificationType (p. 32) operation. Type: String Default: None	No
`SubjectId`	The ID of the Worker requesting the Qualification. This ID corresponds to the `WorkerId` returned with assignment results when the Worker performs a HIT. Type: String Default: None	No
`Test`	The contents of the Qualification test that was presented to the Worker, if the type has a test and the Worker has submitted answers. This value is identical to the QuestionForm associated with the Qualification type at the time the Worker requests the Qualification. Type: a QuestionForm (p. 191) data structure Default: None	No
`Answer`	The Worker's answers for the Qualification type's test contained in a `QuestionFormAnswers` document, if the type has a test and the Worker has submitted answers. If the Worker does not provide any answers, `Answer` may be empty. Type: a QuestionFormAnswers (p. 211) data structure	No

Name	Description	Required
	Default: None	
SubmitTime	The date and time the Qualification request had a status of **Submitted**. This is either the time the Worker submitted answers for a Qualification test, or the time the Worker requested the Qualification if the Qualification type does not have a test. Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **2005-01-31T23:59:59Z** Default: None	No

Example

The following example shows a QualificationRequest data structure returned by the GetQualificationRequests (p. 72) operation. This operation returns the requests for Qualifications of a Qualification type to the owner of the type.

```
<QualificationRequest>
  <QualificationRequestId>789RVWYBAZW00EXAMPLE951RVWYBAZW00EXAMPLE</QualificationRequestId>
  <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
  <SubjectId>AZ3456EXAMPLE</SubjectId>
  <Test>
    &lt;QuestionForm&gt;
      [XML-encoded question data]
    &lt;/QuestionForm&gt;
  </Test>
  <Answer>
    &lt;QuestionFormAnswers&gt;
      [XML-encoded answer data]
    &lt;/QuestionFormAnswers&gt;
  </Answer>
  <SubmitTime>2005-12-01T23:59:59Z</SubmitTime>
</QualificationRequest>
```

QualificationRequirement

Topics

Description

The QualificationRequirement data structure describes a Qualification a Worker must have before the Worker is allowed to accept a HIT. A requirement may optionally state that a Worker must have the Qualification to preview the HIT.

The QualificationRequirement data structure is used as a parameter for the following operations:

- `CreateHIT`
- `RegisterHITType`

The QualificationRequirement data structure is used in the HIT data structure (p. 144).

Using Custom, System-Assigned, and Master Qualification Types

A Qualification requirement can be based on a Qualification you assign to Workers. You can create a custom Qualification type using the `CreateQualificationType` operation. Then you can grant requests for the Qualification automatically using a Qualification test and answer key submitted with the Qualification type, or you can grant the request manually with the `GrantQualification` operation. The `CreateQualificationType` returns a `QualificationTypeId`, which you can use with the `QualificationRequirement` data structure to identify the type of Qualification the Worker is required to have to accept a HIT. Either the Qualification test or your call to `GrantQualification` determines a Qualification value, which is compared to the requirement in the HIT to determine the Worker's eligibility.

Amazon Mechanical Turk supplies several Qualification types for use by all Requesters. Mechanical Turk system-assigned Qualification types work the same way as Qualifications that you create, except that data from the Mechanical Turk marketplace determines the Worker's values. Every Worker has a value for each system Qualification, and these values are updated as the Worker uses the system. Additionally, Amazon Mechanical Turk also provides Master Qualification types that give you access to an elite group of Workers who have demonstrated superior performance while completing thousands of HITs across the Mechanical Turk marketplace. You can use the Master and system-assigned Qualification types by using the corresponding Qualification type ID in the `QualificationTypeId` element of the `QualificationRequirement` data structure. For a list of the Master and system-assigned IDs, see

Qualification Type IDs (p. 165). For more information about using a Master Qualification type, see Master Qualifications (p. 167).

Elements

The QualificationRequirement data structure can contain the following elements.

Name	Description	Value
`QualificationTypeId`	The ID of the Qualification type for the requirement.	A valid QualificationType ID from a custom Qualification type you created or an ID from the list of Master and system-assigned Qualification Type IDs (p. 165).
`Comparator`	The kind of comparison to make against a Qualification's value. You can compare a Qualification's value: • To an `IntegerValue` to see if it is `LessThan`, `LessThanOrEqualTo`, `GreaterThan`, `GreaterThanOrEqualTo`, `EqualTo`, or `NotEqualTo` the `IntegerValue`. • To a `LocaleValue` to see if it is `EqualTo`, or `NotEqualTo` the `LocaleValue`. • To see if the value is `In` or `NotIn` a set of `IntegerValue` or `LocaleValue` values. A Qualification requirement can also test if a Qualification `Exists` or `DoesNotExist` in the user's profile, regardless of its value.	`LessThan` | `LessThanOrEqualTo` | `GreaterThan` | `GreaterThanOrEqualTo` | `EqualTo` | `NotEqualTo` | `Exists` | `DoesNotExist` | `In` | `NotIn`
`IntegerValue`	The integer value to compare against the Qualification's value. `IntegerValue` must not be present if `Comparator` is `Exists` or `DoesNotExist`. `IntegerValue` can only be used if the Qualification type has an integer value; it cannot be used with the `Worker_Locale` QualificationType ID, see Qualification Type IDs (p. 165). When performing a set comparison by using the `In` or the `NotIn` comparator, you can use up to 15 `IntegerValue` elements in a QualificationRequirement data structure. For an example of a set comparison, see Example—Using the QualificationRequirement Data Structure for Comparing Multiple Values (p. 170).	An integer.

Name	Description	Value
`LocaleValue`	The locale value to compare against the Qualification's value. The local value must be a valid ISO 3166 country code or supports ISO 3166-2 subdivisions. `LocaleValue` can only be used with a `Worker_Locale` QualificationType ID, see Qualification Type IDs (p. 165). `LocaleValue` can only be used with the `EqualTo`, `NotEqualTo`, `In`, and `NotIn` comparators. You must only use a single `LocaleValue` element when using the `EqualTo` or `NotEqualTo` comparators. When performing a set comparison by using the `In` or the `NotIn` comparator, you can use up to 30 `LocaleValue` elements in a QualificationRequirement data structure. For an example of a set comparison, see Example —Using the QualificationRequirement Data Structure for Comparing Multiple Values (p. 170).	A Locale (p. 155) data structure.
`RequiredToPreview`	If **`true`**, the question data for the HIT will not be shown when a Worker whose Qualifications do not meet this requirement tries to preview the HIT. That is, a Worker's Qualifications must meet all of the requirements for which `RequiredToPreview` is **`true`** in order to preview the HIT. If a Worker meets all of the requirements where `RequiredToPreview` is **`true`** (or if there are no such requirements), but does not meet all of the requirements for the HIT, the Worker will be allowed to preview the HIT's question data, but will not be allowed to accept and complete the HIT. The default is **`false`**.	A Boolean value, **`true`** or **`false`**

Qualification Type IDs

The following table lists the Master and system-assigned Qualification Type IDs that can be used in the `QualificationTypeId` element.

Name	QualificationTypeId	Description
Masters	Sandbox: 2ARFPLSP75KLA8M8DH1HTEQVJT3SY6	Masters are an elite group of Workers, who have demonstrated superior performance while completing

Name	QualificationTypeId	Description
	Production: 2F1QJWKUDD8XADTFD2Q0G6UTO95ALH	thousands of HITs across the Mechanical Turk marketplace. Masters must maintain this high level of performance or they may lose this distinction. Set the `comparator` parameter to "Exists" to require that Workers have this Qualification. Note that for this Qualification type ID, the `QualificationTypeId` value for the Mechanical Turk Sandbox environment is different than the value for the production environment.
Worker_ NumberHITsApproved	00000000000000000040	Specifies the total number of HITs submitted by a Worker that have been approved. The value is an integer greater than or equal to 0.
Worker_Locale	00000000000000000071	The location of the Worker, as specified in the Worker's mailing address. For more information about the locale Qualification, see the The Locale Qualification (p. 168) section.
Worker_Adult	00000000000000000060	Requires workers to acknowledge that they are over 18 and that they agree to work on potentially offensive content. The value type is boolean, 1 (required), 0 (not required, the default).

Name	QualificationTypeId	Description
Worker_ PercentAssignmentsApproved	00000000000000000L0	The percentage of assignments the Worker has submitted that were subsequently approved by the Requester, over all assignments the Worker has submitted. The value is an integer between 0 and 100. Note that a Worker's approval rate is statistically meaningless for small numbers of assignments, since a single rejection can reduce the approval rate by many percentage points. So to ensure that a new Worker's approval rate is unaffected by these statistically meaningless changes, if a Worker has submitted less than 100 assignments, the Worker's approval rate in the system is 100%. To prevent Workers who have less than 100 approved assignments from working on your HIT, set the Worker_ NumberHITsApproved qualification type ID to a value greater than 100.

Master Qualifications

You can require that Workers must have a Master Qualification to complete your HITs.

To create a Qualification requirement for Masters, specify:

- A `QualificationTypeId` of 2F1QJWKUDD8XADTFD2Q0G6UTO95ALH
- A `Comparator` of `Exists`

> **Note**
> The Master Qualification Type ID values used for the `QualificationTypeId` parameter in the preceding procedures are for the production environment. The ID values to use in the Mechanical Turk Sandbox environment are listed in the Qualification Type IDs (p. 165) table.

Adding Adult Content

Adult content can be offensive to some people. For that reason, if your HIT is adult-oriented, we ask you to use the following procedure.

Adding Adult HITs

1. In the HIT title, include the words "adult content."
2. Specify the worker's qualifications in one of the following ways:

Using the API:

• Set the `CreateHit` parameter, `QualificationRequirement`, to the qualification type, 00000000000000000060. • Set `comparator` parameter to "EqualTo." • Set the `IntegerValue` parameter to 1 (required).

Using the command line tools, in the HIT properties file:
• Set `qualificationTypeId` to 00000000000000000060.
• Set `comparator`, to "EqualTo."
• Set the `IntegerValue` to 1 (required). For example,
`<QualificationRequirement>`
`  <QualificationTypeId>00000000000000000060</QualificationTypeId>`
`  <Comparator>EqualTo</Comparator>`
`  <IntegerValue>1</IntegerValue>`
`</QualificationRequirement>`

3. Define the HIT to be private or previewed.

   This setting prevents anyone who does not qualify from seeing the HIT. To make the HIT private, use one of the following methods:

Using the API, set the `RequiredToPreview` parameter to true.
Using the command line tools, in the HIT properties file, set the private parameter, `qualification.private`, to TRUE.

The Locale Qualification

You can create a Qualification requirement based on the Worker's location. The Worker's location is specified by the Worker to Amazon Mechanical Turk when the Worker creates his account.

To create a Qualification requirement based on the Worker's location, specify:

- A `QualificationTypeId` of 00000000000000000071
- A `Comparator` of `EqualTo` or `NotEqualTo`
- A `LocaleValue` that corresponds to the desired locale

To create a Qualification requirement based on the Worker being in or not in one of several locations, specify:

- A `QualificationTypeId` of 00000000000000000071
- A `Comparator` of `In` or `NotIn`
- Multiple `LocaleValue` values that correspond to the desired locales.

For more information on the format of a `LocaleValue` element, see Locale data structure (p. 155).

Example

The following example shows a QualificationRequirement specifying Workers located in the state of Pennsylvania. Workers located in the state of New York (US-NY) or in the country of India (IN) do not qualify for the HITs with this QualificationRequirement.

```
<QualificationRequirement>
  <QualificationTypeId>00000000000000000071</QualificationTypeId>
  <Comparator>EqualTo</Comparator>
  <LocaleValue>
  <Country>US</Country>
  <Subdivision>PA</Subdivision>
  </LocaleValue>
  </QualificationRequirement>
```

The following example shows a QualificationRequirement specifying Workers not located in the state of California. Workers located in the state of New York or in the country of India (IN) will be qualified to do HITs with this QualificationRequirement.

```
<QualificationRequirement>
  <QualificationTypeId>00000000000000000071</QualificationTypeId>
  <Comparator>NotEqualTo</Comparator>
  <LocaleValue>
  <Country>US</Country>
  <Subdivision>CA</Subdivision>
  </LocaleValue>
  </QualificationRequirement>
```

Example—Using the QualificationRequirement Data Structure

The following example of a QualificationRequirement data structure could be passed in to a call to `CreateHIT`. `CreateHIT` accepts parameters that describe the HIT being created, including one or more Qualification requirements.

Example SOAP Request

In a SOAP request, the QualificationRequirement data structure is specified as the `QualificationRequirement` parameter in XML:

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

Example REST Request

In a REST request, the components of the QualificationRequirement data structure are specified as separate parameters. To specify more than one `QualificationRequirement` in a REST request, increment the sequence number in the parameter name for each value:

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
[...]
&QualificationRequirement.1.QualificationTypeId=789RVWYBAZW00EXAMPLE
&QualificationRequirement.1.Comparator=GreaterThan
&QualificationRequirement.1.IntegerValue=18
&QualificationRequirement.2.QualificationTypeId=ZSPJXD4F1SFZP7YNJWR0
&QualificationRequirement.2.Comparator=EqualTo
&QualificationRequirement.2.IntegerValue=1
```

Example—Using the QualificationRequirement Data Structure for Comparing Multiple Values

The following example of a QualificationRequirement data structure could be passed in to a call to `CreateHIT`. `CreateHIT` accepts parameters that describe the HIT being created, including one or more Qualification requirements.

Example SOAP Request

The following example shows a QualificationRequirement data structure used in a SOAP request that uses multiple `IntegerValue` values when performing a set comparison by using the `In` comparator.

```
<QualificationRequirement>
  <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
  <Comparator>In</Comparator>
  <IntegerValue>10</IntegerValue>
  <IntegerValue>20</IntegerValue>
  <IntegerValue>30</IntegerValue>
</QualificationRequirement>
```

Example SOAP Response

The following is an example of a SOAP QualificationRequirement data structure in a response to the preceding example request.

```
<QualificationRequirement>
  <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
  <Comparator>In</Comparator>
  <IntegerValue>10</IntegerValue>
  <IntegerValue>20</IntegerValue>
  <IntegerValue>30</IntegerValue>
  <RequiredToPreview>0</RequiredToPreview>
</QualificationRequirement>
```

Example REST Request

In a REST request, the components of the QualificationRequirement data structure are specified as separate parameters. The following example shows how to perform set comparisons by using the `In` comparator with multiple `IntegerValue` and `LocaleValue` values.

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
[...]
&QualificationRequirement.1.QualificationTypeId=789RVWYBAZW00EXAMPLE
&QualificationRequirement.1.Comparator=In
&QualificationRequirement.1.IntegerValue.1=10
&QualificationRequirement.1.IntegerValue.2=20
&QualificationRequirement.1.IntegerValue.3=30
&QualificationRequirement.2.QualificationTypeId=ZSPJXD4F1SFZP7YNJWR0
&QualificationRequirement.2.Comparator=In
&QualificationRequirement.2.LocaleValue.1.Country=US
&QualificationRequirement.2.LocaleValue.2.Country=CA
&QualificationRequirement.2.LocaleValue.3.Country=MX
```

QualificationType

Description

The QualificationType data structure represents a Qualification type, a description of a property of a Worker that must match the requirements of a HIT for the Worker to be able to accept the HIT. The type also describes how a Worker can obtain a Qualification of that type, such as through a Qualification test.

The QualificationType data structure is used as a response element for the following operations:

- CreateQualificationType (p. 32)
- GetQualificationType (p. 77)
- SearchQualificationTypes (p. 121)
- UpdateQualificationType (p. 136)

Elements

The QualificationType structure can contain the elements described in the following table:

Name	Description	Required
`QualificationTypeId`	A unique identifier for the Qualification type. A Qualification type is given a Qualification type ID when you call the CreateQualificationType (p. 32) operation operation, and it retains that ID forever. Type: String Default: None	No
`CreationTime`	The date and time the Qualification type was created Type: a dateTime structure in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as **`2005-01-31T23:59:59Z`**. Default: None	No
`Name`	The name of the Qualification type. The type name is used to identify the type, and to find the type using a Qualification type search. Type: String Default: None	No
`Description`	A long description for the Qualification type. Type: String Default: None	No
`Keywords`	One or more words or phrases that describe theQualification type, separated by commas. The	No

Name	Description	Required
	Keywords make the type easier to find using a search. Type: String Default: None	
`QualificationTypeStatus`	The status of the Qualification type. A Qualification type's status determines if users can apply to receive a Qualification of this type, and if HITs can be created with requirements based on this type. Type: String Valid Values: Active | Inactive Default: None	No
`RetryDelayInSeconds`	The amount of time, in seconds, Workers must wait after taking the Qualification test before they can take it again. Workers can take a Qualification test multiple times if they were not granted the Qualification from a previous attempt, or if the test offers a gradient score and they want a better score. Type: positive integer Default: None. If not specified, retries are disabled and Workers can request a Qualification only once.	No
`Test`	The questions for a Qualification test associated with this Qualification type that a user can take to obtain a Qualification of this type. Type: a QuestionForm (p. 191) data structure. **Note** A Qualification test cannot use an ExternalQuestionQuestionForm (p. 216) like a HIT can. Default: None Constraints: must be specified if `AnswerKey` is present. A Qualification type cannot have both a specified `Test` parameter and an `AutoGranted` value of **`true`**.	No
`TestDurationInSeconds`	The amount of time, in seconds, given to a Worker to complete the Qualification test, beginning from the time the Worker requests the Qualification. Type: positive integer Default: None	No

Name	Description	Required
AnswerKey	The answers to the Qualification test specified in the Test parameter. Type: an AnswerKey (p. 212) data structure. Default: None. If not provided with a test, the Qualification author must process the Qualification request manually.	No
AutoGranted	Specifies that requests for the Qualification type are granted immediately, without prompting the Worker with a Qualification test. Type: Boolean Valid Values: true | false Default: None Constraints: A Qualification type cannot have both a specified Test parameter and an AutoGranted value of **true**.	No
AutoGrantedValue	The Qualification value to use for automatically granted Qualifications, if AutoGranted is true. Type: Integer Default: 1	No
IsRequestable	Specifies whether the Qualification type is one that a user can request through the Amazon Mechanical Turk web site, such as by taking a Qualification test. This value is **false** for Qualifications assigned automatically by the system. Type: Boolean Valid Values: true | false Default: None	No

Example

The following example shows a QualificationType data structure returned by a call to the GetQualificationType (p. 77) operation. The GetQualificationType (p. 77) operation returns a GetQualificationTypeResult element, which contains a QualificationType element.

```
<QualificationType>
  <QualificationTypeId>789RVWYBAZW00EXAMPLE</QualificationTypeId>
  <CreationTime>2005-01-31T23:59:59Z</CreationTime>
  <Name>EnglishWritingAbility</Name>
  <Description>The ability to write and edit text...</Description>
  <Keywords>English, text, write, edit, language</Keywords>
  <QualificationTypeStatus>Active</QualificationTypeStatus>
  <RetryDelayInSeconds>86400</RetryDelayInSeconds>
```

```
  <IsRequestable>true</IsRequestable>
</QualificationType>
```

Notification

Description

The Notification data structure describes a HIT event notification for a HIT type.

The Notification data structure is used as a parameter for the following operations:

- SetHITTypeNotification (p. 129)
- SendTestEventNotification (p. 125)

Note
The latest Amazon Mechanical Turk WSDL includes deprecated notification transport signature protocols for backwards compatibility.

Elements

The Notification structure can contain the elements described in the following table. When the structure is used in a request, elements described as **Required** must be included for the request to succeed.

Name	Description	Required
`Destination`	The destination for notification messages. Type: • For email notifications (if `Transport` is **`Email`**), this is an email address. • For Amazon Simple Queue Service (Amazon SQS) notifications (if `Transport` is **`SQS`**), this is the URL for your Amazon SQS queue. For more information, see Notification Handling Using Amazon SQS (p. 226). Default: None	Yes
`Transport`	The method Amazon Mechanical Turk uses to send the notification. Type: String Valid Values: Email | SQS Default: None	Yes
`Version`	The version of the Notification API WSDL/schema, see WSDL and Schema Locations (p. 2). Type: String Default: None	Yes
`EventType`	The events that should cause notifications to be sent. You can specify multiple events by repeating this element. The Ping event is only valid for the SendTestEventNotification (p. 125) operation.	Yes

Name	Description	Required
	Type: String Valid Values: AssignmentAccepted | AssignmentAbandoned | AssignmentReturned | AssignmentSubmitted | HITReviewable | HITExpired | Ping Default: None	

Example

In the following example the notification specification specifies that an event notification message will be sent by email when a Worker returns or abandons a HIT and the message will use the **2006-05-05** version of the notification message schema.

```
<Notification>
  <Destination>janedoe@example.com</Destination>
  <Transport>Email</Transport>
  <Version>2006-05-05</Version>
  <EventType>AssignmentAbandoned</EventType>
  <EventType>AssignmentReturned</EventType>
</Notification>
```

WorkerBlock

Description

The `WorkerBlock` data structure represents a Worker who has been blocked. It has two elements: the `WorkerId` and the `Reason` for the block.

The `WorkerBlock` data structure is used in the results of the following operation:

- GetBlockedWorkers (p. 56)

Elements

The WorkerBlock structure contains the elements described in the following table.

Name	Description
`WorkerId`	The ID of the Worker who accepted the HIT. Type: String Default: None
`Reason`	A message explaining the reason the Worker was blocked. Type: String Default: None

Example

The following example shows a sample `WorkerBlock` data structure in a response from the GetBlockedWorkers (p. 56) operation.

In a SOAP request, the `WorkerBlock` data structure is specified as the WorkerBlock parameter in XML:

```
<WorkerBlock>
  <WorkerId>AZ3456EXAMPLE</WorkerId>
  <Reason>After several  warnings, he continued to submit answers without reading the
 instructions carefully.</Reason>
</WorkerBlock>
```

In a REST request, the components of the `WorkerBlock` data structure are specified as separate parameters:

```
https://mechanicalturk.amazonaws.com/?Service=AWSMechanicalTurkRequester
[...]
&WorkerBlock.1.WorkerId=AZ3456EXAMPLE
&WorkerBlock.1.Reason=After%20several%20warnings,%20he%20continued%20to%20submit%20answers
%20without%20reading%20the%20instructions%20carefully
```

Review Policies

Using Amazon Mechanical Turk Review Policies you can evaluate Worker submissions against a defined set of criteria. You specify the Review Policy(s) that you want to use when you call the CreateHIT (p. 21) operation.

There are two types of Review Policies, Assignment-level and HIT-level:

- An Assignment-level Review Policy is applied as soon as a Worker submits an assignment. For more information, see Assignment Review Policies (p. 179).
- A HIT-level Review Policy is applied when a HIT becomes reviewable. For more information, see HIT Review Policies (p. 181).

You can select from a set of pre-defined Review Policies. One Review Policy leverages *known answers* or *gold standards* within a Human Intelligence Task (HIT) and has Mechanical Turk calculate a Worker's performance on these known answers. You can specify an action for Mechanical Turk to take automatically based on Worker performance against the known answer.

Mechanical Turk has Review Policies that calculate consensus/agreement among multiple Workers performing the same HITs. For instance, you can specify a Review Policy that measures agreement on work items within the HIT and authorizes Mechanical Turk to keep asking additional Workers to work on the HIT, until a certain level of agreement is achieved. Once the required level of agreement is achieved, the results are returned to you for immediate use.

Review Policies that track Worker performance on your known answers and agreement with other Workers give you information you can use to manage your Workers. For more information about using Review Policies, see Review Policy Use Cases (p. 185).

How Review Policies Work

You specify the Review Policy(s) that you want Mechanical Turk to apply when you call the CreateHIT (p. 21) operation. You must specify Review Policies when you create a HIT. You cannot apply a Review Policy to an existing HIT.

As assignments are submitted, Mechanical Turk applies the Review Policy(s) that you specify. You call the GetReviewResultsForHIT (p. 92) operation to gather the results from the application of the Review Policy.

There are two types of Review Policies, Assignment-level Review Policies that are applied as soon as a Worker submits an assignment and HIT-level Review Policies that are applied when a HIT becomes reviewable. For more information, see Assignment Review Policies (p. 179) and HIT Review Policies (p. 181).

You can specify one Assignment-level Review Policy and one HIT-level Review Policy when you call `CreateHIT` using the HIT Review Policy (p. 151) data structure. The Assignment-level Review Policy ScoreMyKnownAnswer/2011-09-01 and the HIT-level Review Policy SimplePlurality/2011-09-01 can be used in the same call to `CreateHIT`.

Once an Assignment-level Review Policy is applied, the Assignment's status is changed to *Submitted* and optionally an event notification can be sent. Assignments with *Submitted* status are returned by the

GetAssignmentsForHIT (p. 52) operation and the results of applying the Review Policy are available by using the GetReviewResultsForHIT (p. 92) operation.

You can use different Review Policies on distinct HITs in a HIT type. For example, you may wish to apply the ScoreMyKnownAnswers/2011-09-01 policy to a small number of HITs that have known answers in them, but apply the SimplePlurality/2011-09-01 policy to all HITs in a group. Workers do not have access on the Worker User Interface to information about whether a Review Policy has been applied to a HIT.

To help you understand Worker performance on your Review Policy you can call the GetRequesterWorkerStatistic (p. 85) operation to get the percentage of known answers that were answered correctly or the Worker agreement level for your HITs.

Assignment Review Policies

Assignment-level Review Policies are applied as soon as a Worker submits an assignment.

ScoreMyKnownAnswers/2011-09-01

ScoreMyKnownAnswers/2011-09-01 is an Assignment-level Review Policy.

Description

You can use the ScoreMyKnownAnswers/2011-09-01 Review Policy for QuestionForm (QAP) HITs and for ExternalQuestion (iframe) HITs. You provide an answer key when you call the CreateHIT (p. 21) operation. The answer key is a collection of QuestionIds, where each QuestionId has a set of zero or more values that represent the correct response for that QuestionId. For more information about QuestionForm and ExternalQuestion HITs, see QuestionForm (p. 191) and ExternalQuestion (p. 216).

You can specify if one question in your HIT has a known answer or if many questions in your HIT have known answers. When a Worker submits an assignment Mechanical Turk examines the Worker's answers and compares them against the set of known answers that you provide when you create the HIT. Mechanical Turk then calculates a score, for example, 4 out of 10 known answers were correct.

Based on how the Worker's level of agreement with the known answers compares with various configurable thresholds, Mechanical Turk can automatically take actions you requested to approve the assignment, automatically reject the assignment, or automatically extend the HIT to publish an assignment for another Worker.

A Worker's performance on known answers within a specific assignment are returned from calling the GetReviewResultsForHIT (p. 92) operation. You can get a Worker's Life to Date (LTD), 7 day, and 30 day known answer statistics using the GetRequesterWorkerStatistic (p. 85) operation.

Mechanical Turk evaluates answers and considers the following answers as not matching:

- The Worker left an empty value set in the answer key.
- The answer key has an empty value set but the Worker supplied an answer.
- The Worker provides an answer that is the wrong case or has incorrect punctuation that doesn't match the answer exactly. You can either use structured HTML form elements to restrict the values a Worker can submit, or use JavaScript to validate and normalize the submitted values.
- The answer key says a question's answer is A and B but the Worker's value is A.
- The answer key says a question's answer is A and the Worker selected both A and B.

When comparing answers for a match, Mechanical Turk removes any whitespace from before and after the Worker's answer, and from before and after the answer you provide.

Parameters

The following parameters are specified in the AssignmentReviewPolicy element when calling the `CreateHIT` operation. You must also specify the PolicyName *ScoreYourKnownAnswers/2011-09-01* as part of the AssignmentReviewPolicy element. For an example of how to structure the AssignmentReviewPolicy element, see the HIT Review Policy (p. 151) data structure.

Name	Description	Required
`AnswerKey`	Question IDs and the answers to the questions. Type: MapEntry, see the HIT Review Policy (p. 151) data structure. Default: None	Yes
`ApproveIfKnownAnswerScoreIsAtLeast`	Approve the assignment if the KnownAnswerScore is equal to or greater than this value. If not specified, assignments are left in the submitted state and are not approved or rejected. Type: Integer Constraints: Minimum value 0 (always approve), maximum 101 (never approve)	No
`ApproveReason`	A description provided to the Worker about the reason the assignment was approved. If not specified, the reason is left blank. Type: String	No
`RejectIfKnownAnswerScoreIsLessThan`	Reject the assignment if the KnownAnswerScore is equal to or less than this value. If not specified, assignments are left in the submitted state and are not approved or rejected. Type: Integer Constraints: Minimum value 0 (never reject), maximum 101 (always reject).	No
`RejectReason`	A description provided to the Worker about the reason the assignment was rejected. If not specified, the reason is left blank. Type: String	No
`ExtendIfKnownAnswerScoreIsLessThan`	Extend the HIT by one assignment to allow one more Worker to complete it if the known answer score is less than this value. Ordinarily this is done to replace an assignment that is being rejected or that is not usable because	No

Name	Description	Required
	the Worker didn't answer the known answer correctly. If omitted the HIT is not extended. Type: String Constraint: Minimum value 0 (never extend), maximum 101 (always extend).	
`ExtendMaximumAssignments`	The maximum number of assignments the HIT can be extended. Note that if you use the ExtendHIT (p. 42) operation and specify a maximum assignment count greater than this value, ScoreMyKnownAnswers will not extend the HIT. **Note**: If a HIT is created with fewer than 10 assignments, it will not extend to have 10 or more assignments. Type: Integer Constraint: Minimum value 2, maximum 25. Default: 5	No
`ExtendMinimumTimeInSeconds`	The additional time in seconds to let other Workers complete the extended assignment. Type: Integer Constraints: Minimum of 3600 (one hour), Maximum of 31536000 (one year). Default: 0	No

HIT Review Policies

A HIT-level Review Policy is applied when a Human Intelligence Task (HIT) becomes reviewable.

SimplePlurality/2011-09-01

SimplePlurality/2011-09-01 is a HIT-level Review Policy.

Description

The SimplePlurality/2011-09-01 policy allows you to automatically compare answers received from multiple Workers and detect if there is a majority or consensus answer. The results can optionally trigger additional actions, such as approving the assignments that matched the majority answer. The results of this comparison are available as a part of the GetReviewResultsForHIT (p. 92) operation.

Mechanical Turk evaluates answers and considers the following answers as not matching:

- The Worker provides an answer that is the wrong case or incorrect punctuation that doesn't match the answer exactly to another Worker. You can either use structured HTML form elements to restrict the values a Worker can submit, or use JavaScript to validate and normalize the submitted values.
- One Worker's answer is A and B, but another Worker's value is A.
- One Worker's answer is A, but another Worker selected both A and B.

When comparing answers for a match, Mechanical Turk removes any whitespace from before and after the Worker's answer.

> **Note**
> Answers that are longer than 256 characters are not used in the computation of HIT review policies.

Parameters

The following parameters are specified in the HITReviewPolicy element when calling the CreateHIT (p. 21) operation. You must also specify the PolicyName *SimplePlurality/2011-09-01* as part of the HitReviewPolicy element. For an example, see HIT Review Policy (p. 151) data structure.

Name	Description	Required
`QuestionIds`	A comma-separated list of questionIds used to determine agreement. Type: String Constraints: none	Yes
`QuestionAgreementThreshold`	If the Question Agreement Score is greater than this value, the questionId is considered to have an *agreed answer*. Type: Integer Constraints: none	Yes
`DisregardAssignmentIfRejected`	Excludes rejected assignments from agreement calculation. Type: Boolean Constraints: T or F	Yes
`DisregardAssignmentIfKnownAnswerScoreIsLessThan`	Excludes answers from agreement calculation if the KnownAnswerScore is present and less than the provided value. Type: Integer Constraints: none	No
`ExtendIfHITAgreementScoreIsLessThan`	If the HIT Agreement Score is less than this value, extend the HIT to another Worker to complete. If omitted, extending on failure is disabled. Type: Integer	No

Name	Description	Required
	Constraints: 1-100	
ExtendMaximumAssignments	If the ExtendIfHITAgreementScoreIsLessThan is provided, this sets the total maximum number of assignments for the HIT. If you use ExtendHIT operation and specify the maximum assignment count greater than this value, ScoreMyKnownAnswers will not extend the HIT. **Note**: If a HIT is created with fewer than 10 assignments, it will not extend to have 10 or more assignments. Type: Integer Constraints: none Conditions: Required if ExtendIfHITAgreementScoreIsLessThan is provided.	Conditional
ExtendMinimumTimeInSeconds	If the ExtendIfHITAgreementScoreIsLessThan is provided, this sets the additional time that the HIT will be extended by. Type: Integer Constraints: Minimum 3600 (one hour) Maximum 31536000 (365 days) Conditions: Required if ExtendIfHITAgreementScoreIsLessThan is provided.	Conditional
ApproveIfWorkerAgreementScoreIsAtLeast	If the Worker Agreement Score is not less than this value, approve the Worker's assignment. If omitted, assignment will not be approved or rejected. Type: Integer Constraints: none	No
RejectIfWorkerAgreementScoreIsLessThan	If the Worker Agreement Score is less than this value, reject the Worker's assignment. If omitted, assignment will not be approved or rejected. Type: Integer Constraints: none	No

Name	Description	Required
`RejectReason`	If the RejectIfWorkerAgreementIsScoreLessThan value is provided, this value sets the reason for any automated rejections. Type: String Constraints: none	Optional

Scores

The following scores are calculated data from the SimplePlurality/2011-09-01 policy. Based on the value of these scores, Mechanical Turk can take various actions that you specify in the `CreateHIT` operation. It is important to understand how these scores are calculated so you can specify the appropriate actions to take, including approving or rejecting assignments, or extending HITs. The following chart describes how the scores are calculated.

Score	Description
Question Agreement Score	Percentage of Workers who provided the agreed-upon answer for a HIT. Note: Answer values are not normalized for case, whitespace, or punctuation before comparison. Answers can contain multiple values (such as in a set of check boxes); two answers agree with each other if they have the same values present and absent. We don't recommend using free format answers because values are not normalized.
HIT Agreement Score	Percentage of questions within the HIT with an agreed-upon answer. The number of questions within the HIT with an agreed-upon answer, divided by the number of questions evaluated.
Worker Agreement Score	The percentage of questions to which a Worker's answer agreed with other Workers' answers in the same HIT. If a question does not have an agreed upon answer the question is disregarded in this calculation.

The example chart below describes how the Answer Agreement Score and Worker Agreement Score is calculated for a HIT with 4 questions and answers from 3 Workers.

QuestionId	Worker1's answers	Worker2's answers	Worker3's answers	Has Agreed-upon value?	Agreed-upon value	Question Agreement Score
A	coat	sweater	coat	Yes	coat	66%
B	blue	blue	green	Yes	blue	66%
C	large	large	large	Yes	large	100%
D	Furry	fur	furr	No	n/a	n/a
Worker Agreement Score	100%	66%	66%			

The Question Agreement Score for questions A and B are 66% because two Workers agreed on the same answer. The HIT Agreement Score for this HIT is 75%. The HIT had four questions, and three of them had an agreed-upon answer for a percentage of 75%. The Worker Agreement Score for Worker 1 is 100% because this Worker agreed with the other Workers for each answer, except Question D where there was no conclusive answer.

Review Policy Use Cases

The following use cases show you how to apply ScoreYourKnownAnswers and SimplePlurality policies when you call the CreateHIT (p. 21) operation.

Photo Moderation Use Case – Single Worker with Known Answers

In this scenario, you want Workers to moderate photos and screen the photos for inappropriate content. You place 20 photos in a single HIT and 5 of the 20 photos are your known answers. You are using Master Workers and have created the HIT with only one initial assignment. You want to use the answers based on the Worker getting at least 4 of the 5 known answers (80% Answer Agreement Score) correct. If the first Worker does not meet the Answer Agreement score of 80%, then you want to extend the HIT to another Worker. But, in this scenario, you only want to extend the HIT to a maximum of three Workers.

Elements and Parameters

The following is a list of elements and parameters you need to specify in the CreateHIT (p. 21) operation to execute the above scenario and allow Mechanical Turk to automatically calculate the known answer score. Note that this `CreateHIT` example assumes you have already created a HIT Type.

Element	Parameter	Value
`AssignmentReviewPolicy`	PolicyName	ScoreMyKnownAnswers/2011/09/01
`AssignmentReviewPolicy`	AnswerKey	List of questionIDs and answers.
`AssignmentReviewPolicy`	ExtendIfKnownAnswerScoreIsLessThan	80
`AssignmentReviewPolicy`	ExtendMaximumAssignments	3

Examples

The following example shows how to use the above elements and parameters with the `CreateHIT` operation.

Sample CreateHIT Request

The following example shows a `CreateHIT` request.

```
<CreateHITRequest>
    <HITTypeId>T100CN9P324W00EXAMPLE</HITTypeId>
    <Question>[CDATA block or XML Entity encoded]</Question>
```

```
    <LifetimeInSeconds>604800</LifetimeInSeconds>
    <AssignmentReviewPolicy>
        <PolicyName>ScoreMyKnownAnswers/2011-09-01</PolicyName>
        <Parameter>
            <Key>AnswerKey</Key>
            <MapEntry>
                <Key>QuestionId3</Key>   <!—correct answer is “B” -->
                <Value>B</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId7</Key>  <!—correct answer is “A” -->
                <Value>A</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId15</Key>   <!—correct answer is “F” -->
                <Value>F</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId17</Key>   <!—correct answer is “C” -->
                <Value>C</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId18</Key>   <!—correct answer is “A” -->
                <Value>A</Value>
            </MapEntry>
        </Parameter>
        <Parameter>
           <Key>ExtendIfKnownAnswerScoreIsLessThan</Key>
           <Value>80</Value>
        </Parameter>
        <Parameter>
           <Key>ExtendMaximumAssignments</Key>
           <Value>3</Value>
        </Parameter>
    </AssignmentReviewPolicy>
</CreateHITRequest>
```

Photo Moderation Use Case – Multiple Workers with Agreement

In this scenario, you want Workers to moderate photos and screen the photos for inappropriate content. You place 20 photos in a single HIT and 5 of the 20 photos are your known answers. You want to approve the assignment if the Worker completes at least 4 of the 5 known answers correct (at least 80% Answer Agreement Score).

You want 3 Workers to complete each HIT and you want to calculate the HIT Agreement Score for the 15 photos you don’t know the answer to. Also, you want to disregard the Worker's answer in the Agreement Score if they don't get 4 of 5 of the known answers correct.

Elements and Parameters

The following is a list of elements and parameters you need to specify in the CreateHIT (p. 21) operation to execute the above scenario and allow Mechanical Turk to automatically approve the assignments. Note that this CreateHIT example assumes you have already created a HIT Type.

Element	Parameter	Value
AssignmentReviewPolicy	PolicyName	ScoreMyKnownAnswers/2011/09/01

Element	Parameter	Value
AssignmentReviewPolicy	Answer	List of questionIDs and answers.
AssignmentReviewPolicy	ApproveIfKnownAnswerScoreIsAtLeast	80
AssignmentReviewPolicy	ExtendIfKnownAnswerScoreIsLessThan	80
AssignmentReviewPolicy	ExtendMaximumAssignments	3
HITReviewPolicy	PolicyName	SimplePlurality/2011-09-01
HITReviewPolicy	QuestionIDs	Your list of 15 question IDs.
HITReviewPolicy	QuestionAgreementThreshold	100
HITReviewPolicy	DisregardAssignmentIfKnownAnswerScoreIsLessThan	80

Examples

The following example shows how to use the above elements and parameters with the CreateHIT operation.

Sample CreateHIT Request

The following example shows a CreateHIT request.

```
<CreateHITRequest>
    <HITTypeId>T100CN9P324W00EXAMPLE</HITTypeId>
    <Question>[CDATA block or XML Entity encoded]</Question>
    <LifetimeInSeconds>604800</LifetimeInSeconds>
    <AssignmentReviewPolicy>
        <PolicyName>ScoreMyKnownAnswers/2011-09-01</PolicyName>
        <Parameter>
            <Key>AnswerKey</Key>
            <MapEntry>
                <Key>QuestionId3</Key>   <!—correct answer is “B” -->
                <Value>B</Value>
            </MapEntry
            <MapEntry>
                <Key>QuestionId4</Key>  <!—correct answer is “A” -->
                <Value>A</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId13</Key>   <!—correct answer is “F” -->
                <Value>F</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId14</Key>   <!—correct answer is “C” -->
                <Value>C</Value>
            </MapEntry>
            <MapEntry>
                <Key>QuestionId19</Key>   <!—correct answer is “A” -->
                <Value>A</Value>
            </MapEntry>
         </Parameter>
        <Parameter>
           <Key>ApproveIfKnownAnswerScoreIsAtLeast</Key>
           <Value>80</Value>
        </Parameter>
```

```
        <Parameter>
           <Key>ExtendIfKnownAnswerScoreIsLessThan</Key>
           <Value>80</Value>
        </Parameter>
        <Parameter>
           <Key>ExtendMaximumAssignments</Key>
           <Value>3</Value>
        </Parameter>
    </AssignmentReviewPolicy>
    <HITReviewPolicy>
        <PolicyName>SimplePlurality/2011-09-01</PolicyName>
        <Parameter>
           <Key>QuestionIDs</Key>
           <Value>questionid1</Value>
           <Value>questionid2</Value>
           <Value>questionid5</Value>
           <Value>questionid6</Value>
           <Value>questionid7</Value>
                 ..... <! Add your additional 10 questionIDs for a total of 15 questions.
 Different from your known answer questionIDs.>
        </Parameter>
        <Parameter>
           <Key>QuestionAgreementThreshold</Key>
           <Value>100</Value>
        </Parameter>
        <Parameter>
           <Key>DisregardAssignmentIfKnownAnswerScoreIsLessThan</Key>
           <Value>80</Value>
        </Parameter>
    </HITReviewPolicy>
</CreateHITRequest>
```

Categorization and Tagging Use Case – Multiple Workers

In this scenario, you want Workers to categorize a product and provide multiple tags for the product in a HIT. You also want the Workers to be able to comment on your HIT and give you feedback.

You want to calculate the Answer Agreement Score for only the categorization question. If two Workers do not agree on the product categorization question, you want to extend the HIT to a third Worker. Also, you want to extend the assignment by an hour so the third Worker has time to work on the assignment.

Elements and Parameters

The following is a list of elements and parameters you need to specify in the CreateHIT (p. 21) operation to execute the above scenario and allow Mechanical Turk to automatically calculate agreement and approve or reject the assignments. Note that this CreateHIT example assumes you have already created a HIT Type.

Element	Parameter	Value
`HITReviewPolicy`	PolicyName	SimplePlurality/2011-09-01
`HITReviewPolicy`	QuestionIDs	questionID1
`HITReviewPolicy`	QuestionAgreementThreshold	100
`HITReviewPolicy`	ExtendMinimumTimeInSeconds	3600

Element	Parameter	Value
HITReviewPolicy	ExtendMaximumAssignments	3

Examples

The following example shows how to use the above elements and parameters with the CreateHIT operation.

Sample CreateHIT Request

The following example shows a CreateHIT request.

```
<CreateHITRequest>
    <HITTypeId>T100CN9P324W00EXAMPLE</HITTypeId>
    <Question>[CDATA block or XML Entity encoded]</Question>
    <LifetimeInSeconds>604800</LifetimeInSeconds>
    <HITReviewPolicy>
        <PolicyName>SimplePlurality/2011-09-01</PolicyName>
        <Parameter>
           <Key>QuestionIDs</Key>
           <Value>questionID1</Value>
        </Parameter>
        <Parameter>
           <Key>QuestionAgreementThreshold</Key>
           <Value>100</Value>
        </Parameter>
        <Parameter>
           <Key>ExtendMaximumAssignments</Key>
           <Value>3</Value>
        </Parameter>
        <Parameter>
           <Key>ExtendMinimumTimeInSeconds</Key>
           <Value>3600</Value>
        </Parameter>
    </HITReviewPolicy>
</CreateHITRequest>
```

Question and Answer Data

Topics

The questions and answers that Amazon Mechanical Turk passes between Requesters and Workers are XML documents that conform to schemas. These documents are passed to the service and returned by the service as parameter values.

Using XML Parameter Values

The QuestionForm (p. 191), QuestionFormAnswers (p. 211), and AnswerKey (p. 212) data structures are used as parameter values in service requests, and as return values in service responses. Unlike other data structures described in this API reference, these XML structures are not part of the service API directly, but rather are used as string values going in and out of the service. This article describes the encoding methods needed to use XML data as parameter and return values.

XML Data as a Parameter

For SOAP requests, XML data in a parameter value must appear in the request *XML escaped*. Characters that are part of XML syntax, such as ampersands (**&**) and angle brackets (**<>**), must be replaced with the corresponding XML character entities in the parameter value. Most SOAP toolkits will automatically escape data set as the string value of the parameter.

The following is a fragment of a `QuestionForm` data structure, escaped with XML character entities:

```
&lt;QuestionForm xmlns="..."&gt;
  &lt;Overview&gt;
    &lt;Text&gt;
      Musicals by Rodgers &amp;amp; Hart...
    &lt;/Text&gt;
  &lt;Overview&gt;
  ...
&lt;/QuestionForm&gt;
```

For REST requests, the data must be *URL encoded* to appear as a single parameter value in the request. (This is true for all REST parameter values.) Characters that are part of URL syntax, such as question marks (**?**) and ampersands (**&**), must be replaced with the corresponding URL character codes.

> **Note**
> XML data in REST requests should only be URL encoded, *not* XML escaped.

In service responses, this data will be XML escaped.

Namespaces for XML Parameter Values

XML data in parameter values must have a namespace specified for all elements. The easiest way to do this is to include an `xmlns` attribute in the root element equal to the appropriate namespace.

The namespace for a `QuestionForm`, `QuestionFormAnswers`, or `AnswerKey` element is identical to the URL of the corresponding schema document, including the version date. While XML namespaces need not be URLs according to the XML specification, this convention ensures that the consumer of the value knows which version of the schema is being used for the data.

For the locations of the schema documents, as well as instructions on how to include the version date in the URL, see WSDL and Schema Locations (p. 2).

QuestionForm

Topics

- Answer Specification (p. 199)
- Example (p. 205)

Description

The `QuestionForm` data format describes one or more *questions* for a HIT, or for a Qualification test. It contains instructions and data Workers use to answer the questions, and a set of one or more form fields, which are rendered as a web form for a Worker to fill out and submit.

A `QuestionForm` is a string value that consists of XML data. This XML data must conform to the `QuestionForm` schema. All elements in a `QuestionForm` belong to a namespace whose name is identical to the URL of the `QuestionForm` schema document. See WSDL and Schema Locations (p. 2) for the location of this schema.

> **Tip**
> For information about creating HITs that use your own web site in a frame instead of questions, see the ExternalQuestion data structure (p. 216).

The `QuestionForm` data structure is a value in a HIT data structure (p. 144) and a value in a QualificationType data structure (p. 171). The `QuestionForm` data structure is used as a parameter value for the following operations:

- `CreateHIT`
- `CreateQualificationType`
- `UpdateQualificationType`

For more information about using XML data as a parameter or return value, see Using XML Parameter Values (p. 191).

QuestionForm Structure

The top-most element of the `QuestionForm` data structure is a `QuestionForm` element. This element contains optional `Overview` elements and one or more `Question` elements. There can be any number of these two element types listed in any order. The following example structure has an `Overview` element and a `Question` element followed by a second `Overview` element and `Question` element—all within the same `QuestionForm`.

```
<QuestionForm xmlns="[the QuestionForm schema URL]">
  <Overview>
    [...]
  </Overview>
  <Question>
    [...]
  </Question>
  <Overview>
    [...]
  </Overview>
  <Question>
    [...]
  </Question>
  [...]
</QuestionForm>
```

The `Overview` element describes instructions and information, and presents them separately from the set of questions. It can contain any kind of informational content, as described below. If omitted, no overview text is displayed above the questions.

Each `Question` element can contain the elements described in the following table. See also the example below the table.

Name	Description	Required
`QuestionIdentifier`	An identifier for the question. This identifier is used to associate the Worker's answers with the question in the answer data. Type: String Default: None	Yes
`DisplayName`	A name for the question, displayed as a prominent heading. Type: String Default: None	No
`IsRequired`	Specifies whether the Worker must provide an answer for this question to successfully submit the form. Type: Boolean Default: false Valid Values: true | false	No
`QuestionContent`	The instructions and data specific to this question, such as the text of the question. It can contain any kind of informational content, as described in the *Content Structure* section below. Type: Content structure Default: None	Yes
`AnswerSpecification`	A structure that describes the field type and possible values for the answer to this question, as described in the *Answer Specification* section below. This element controls how the form field is rendered and specifies which values are valid answers for this question. Type: An answer specification structure Default: None Valid Values: `FreeTextAnswer` | `SelectionAnswer` | `FileUploadAnswer`	Yes

For example:

```
<Question>
  <QuestionIdentifier>my_question_id</QuestionIdentifier>
  <DisplayName>My Question</DisplayName>
  <IsRequired>true</IsRequired>
  <QuestionContent>
    [...]
```

```
    </QuestionContent>
    <AnswerSpecification>
      [...]
    </AnswerSpecification>
  </Question>
```

Content Structure

The `Overview` elements and the `QuestionContent` elements of a `QuestionForm` can contain different types of information. For example, you might include a paragraph of text and an image in your HIT's overview.

Each kind of information is defined by a corresponding element. These elements can appear in any number, in any order. The content elements are rendered in the order in which they occur in the containing element.

Following are the allowed information types:

- Title
- Text
- List
- Binary
- Application
- EmbeddedBinary
- FormattedContent

Each of these types are described in detail in the following subsections. A full example showing the use of the elements and information types is at the end of the section.

Title

A `Title` element specifies a string to be rendered as a title or heading.

```
<Title>The Next Move</Title>
```

Text

A `Text` element specifies a block of text to be rendered as a paragraph. Only plain text is allowed. HTML is not allowed. If HTML characters (such as angle brackets) are included in the data, they appear verbatim in the web output.

```
<Text>What is the best next move for "X" in this game of Tic-Tac-Toe?</Text>
```

List

A `List` element displays a bulleted list of items. Items are specified using one or more `ListItem` elements inside the `List`. The `ListItem` element is a string.

```
<List>
  <ListItem>It must be a valid move.</ListItem>
  <ListItem>"X" cannot resign.</ListItem>
</List>
```

Binary

A `Binary` element specifies non-textual data of some kind, such as an image, audio, or video. The elements listed in the following table are required and must be entered in the order shown here.

Name	Description	Required
`MimeType`	Specifies the type of the data. Type: `MimeType` element Default: None Child Elements: • A required string that specifies the type of the data. The possible values are **`image`**, **`audio`**, or **`video`**. • An optional string that specifies the format of the item, such as **`gif`**	Yes
`DataURL`	The data itself specified with a `DataURL` element that contains a valid HTTP URL. Type: `DataURL` element Default: None	Yes
`AltText`	The text that should appear if the data cannot be rendered in the browser. Type: String Default: None	Yes

```
<Binary>
  <MimeType>
    <Type>image</Type>
    <SubType>gif</SubType>
  </MimeType>
  <DataURL>http://tictactoe.amazon.com/game/01523/board.gif</DataURL>
  <AltText>The game board, with "X" to move.</AltText>
</Binary>
```

Application

An `Application` element specifies an embedded application. It contains either a `JavaApplet` element or a `Flash` element.

You can specify zero or more parameters to pass to your Java applet or Flash application when it is opened in the web page. For a HIT, in addition to the parameters you specify, Amazon Mechanical Turk includes two parameters specific to the HIT: `hitId` and `assignmentId`. The `hitId` parameter is equal to the ID of the HIT. The `assignmentId` parameter is equal to the ID of the assignment if the Worker has accepted the HIT, or equal to `ASSIGNMENT_ID_NOT_AVAILABLE` if the Worker is only previewing the HIT.

The `JavaApplet` element includes the elements described in the following table:

Name	Description	Required
`AppletPath`	The URL path to the directory that contains Java classes for the applet. Type: URL Default: None	Yes
`AppletFilename`	The name of the class file that contains the applet code, which is located in the path specified by `AppletPath`. Type: String Default: None	Yes
`Width`	The width of the bounding box for the applet. Type: String Default: None	Yes
`Height`	The height of the bounding box for the applet. Type: String Default: None	Yes
`ApplicationParameter`	The parameters for the applet. Type: `ApplicationParameter` Default: None Child Elements: • A required string that specifies the name of the parameter • A required string that specifies the value of the parameter	No

The `Flash` element includes the elements described in the following table:

Name	Description	Required
`FlashMovieURL`	The URL of the Flash movie file. Type: URL Default: None	Yes
`Width`	The width of the bounding box for the Flash movie. Type: String Default: None	Yes

Name	Description	Required
Height	The height of the bounding box for the Flash movie. Type: String Default: None	Yes
ApplicationParameter	The parameters for the Flash movie. Type: ApplicationParameter Default: None Child Elements: • A required string that specifies the name of the parameter • A required string that specifies the value of the parameter	No

```
<Application>
  <JavaApplet>
    <AppletPath>http://tictactoe.amazon.com/applets/</AppletPath>
    <AppletFilename>GameViewer.class</AppletFilename>
    <Width>400</Width>
    <Height>300</Height>
    <ApplicationParameter>
      <Name>game_id<Name>
      <Value>01523</Value>
    </ApplicationParameter>
  </JavaApplet>
</Application>
```

EmbeddedBinary

An EmbeddedBinary element specifies an external object of non-textual data of some kind, such as an image, audio or video, that displays in your browser. The elements listed in the following table are required and must be entered in the order shown here.

Name	Description	Required
EmbeddedMimeType	Specifies the type of the data. Type: EmbeddedMimeType element Default: None Child Elements: • A required string that specifies the type of the data. The possible values are **image**, **audio**, or **video**. • An optional string that specifies the format of the item, such as **gif**	Yes
DataURL	The data itself specified by a DataURL element that contains a valid HTTP URL	Yes

Name	Description	Required
	Type: `DataURL` element Default: None	
`AltText`	The text that should appear if the data cannot be rendered in the browser. Type: String Default: None	Yes
`Width`	The width of the bounding box for the object. Type: String Default: None	Yes
`Height`	The height of the bounding box for the object. Type: String Default: None	Yes
`ApplicationParameter`	The parameters for the `EmbeddedBinary` object. Type: `ApplicationParameter` Default: None Child elements: • A required string that specifies the name of the parameter • A required string that specifies the value of the parameter	No

```
<EmbeddedBinary>
  <EmbeddedMimeType>
    <Type>image</Type>
    <SubType>gif</SubType>
  </EmbeddedMimeType>
  <DataURL>http://tictactoe.amazon.com/game/01523/board.gif</DataURL>
  <AltText>The game board, with "X" to move.</AltText>
  <Width>400</Width>
  <Height>300</Height>
  <ApplicationParameter>
    <Name>game_id<Name>
    <Value>01523</Value>
  </ApplicationParameter>
</EmbeddedBinary>
```

FormattedContent

For finer control over the display of your HIT information, you can specify a `FormattedContent` element. Formatted content is a block of text with formatting information specified using XHTML tags. For example, you can use XHTML tags to specify that certain words appear in a boldface font or to include a table in your HIT information.

Only a limited subset of XHTML is supported. For more information on the creating and validating XHTML formatted content, see Formatted Content: XHTML (p. 206).

The value of the `FormattedContent` element must be specified as an XML CDATA block. CDATA tells the web service that the XHTML elements are not part of the `QuestionForm` data schema. For example, the following describes a paragraph of formatted text:

```
<FormattedContent><![CDATA[
  <p>This is a paragraph with <b>bold text</b>,
  <i>italic text</i>, and <b><i>bold italic text</i></b>.</p>
]]></FormattedContent>
```

Answer Specification

The `AnswerSpecification` element describes the format and possible values for answers to a question. It contains a `FreeTextAnswer` element, which describes a text field; a `SelectionAnswer` element, which describes a multiple choice field; or a `FileUploadAnswer`, which prompts the Worker to upload a file as the answer to the question.

FreeTextAnswer

A `FreeTextAnswer` element describes a text field and constraints on its possible values. It includes the elements described in the following table:

Name	Description	Required
`Constraints`	Describes the constraints on the allowed values for the text field. This element is described in the next table. Type: `Constraints` element Default: None	No
`DefaultText`	Specifies default text. This value appears in the form when it is rendered, and is accepted as the answer if the Worker does not change it. Type: String Default: An empty value	No
`NumberOfLinesSuggestion`	Specifies how tall the form field should be, if possible. The field might be rendered as a text box with this many lines, depending on the device the Worker is using to see the form. Type: Integer Default: 1	No

Note
A Qualification test that is to be graded automatically using an answer key cannot have any free-text questions. An answer key can only match multiple-choice questions and cannot match free-text fields.

The optional `Constraints` element describes constraints on the allowed values for the text field. If no constraints are specified, any value is accepted for the field.

The `Constraints` element includes the elements described in the following table:

Name	Description	Required
`IsNumeric`	Specifies that the value entered must be numeric. Type: empty element Default: None Attributes: • An optional integer that specifies the minimum value allowed • An optional integer that specifies the maximum value allowed	No
`Length`	Specifies the length range of the answer. Type: empty element Default: None Attributes: • An optional non-negative integer that specifies the minimum number of characters • An optional positive integer that specifies the maximum number of characters	No
`AnswerFormatRegex`	Specifies that JavaScript validates the answer string against a given pattern. **Note** A limitation of this approach is that Workers who have disabled JavaScript on their browsers cannot validate their answers. Although this is uncommon, you might want to caution your Workers. Type: empty element Default: None Attributes: • A required string that specifies the regular expression that JavaScript uses to validate against the Workers' entered values • An optional string that allows you to edit the content of errors displayed to the Worker on the Worker web site if the regex validation fails. If this attribute is not specified, the error displayed is "Invalid input supplied." • An optional string with the value `i` which specifies that case is ignored when matching characters	No

The `Constraints` element can contain multiple `AnswerFormatRegex` elements. All `AnswerFormatRegex` constraints must be satisfied before the Worker can submit the HIT.

The following examples demonstrate how to use the `FreeTextAnswer` element.

If you want a 3-digit positive integer between 100 and 999, use the following:

```
<FreeTextAnswer>
  <Constraints>
    <IsNumeric minValue="100" maxValue="999"/>
    <Length minLength="3" maxLength="3"/>
  </Constraints>
</FreeTextAnswer>
```

If you want a 3-digit number that includes decimals, use the following:

```
<FreeTextAnswer>
  <Constraints>
     <IsNumeric/>
     <Length minLength="3" maxLength="3"/>
  </Constraints>
</FreeTextAnswer>
```

If you want to ensure that there is some text, use the following example. The **`minLength`** attribute includes whitespaces in the character count.

```
<FreeTextAnswer>
  <Constraints>
    <Length minLength="2" />
    <AnswerFormatRegex regex="\S" errorText="The content cannot be blank."/>
  </Constraints>
</FreeTextAnswer>
```

If you specify the **`minLength`** attribute, it is the same as if the `IsRequired` element is `true`. If you want to allow an *optional* string that must be at least two characters, use the following:

```
<FreeTextAnswer>
  <Constraints>
    <AnswerFormatRegex regex="(^$|\S{2,})"
     errorText="You must enter at least two characters."/>
  </Constraints>
</FreeTextAnswer>
```

To request a US phone number in the format 1-nnn-nnn-nnnn, where "1-" is optional, use the following:

```
<FreeTextAnswer>
  <Constraints>
    <AnswerFormatRegex
     regex="^(1[- ]?)?(\([2-9]\d{2}\)\s*|[2-9]\d{2}-?)[2-9]\d{2}-?\d{4}$)"
     errorText="You must enter a US phone number in the format
     1-555-555-1234 or 555-555-1234."/>
  </Constraints>
</FreeTextAnswer>
```

If you want an answer that contains a date formatted as yyyy-mm-dd, use the following:

```
<FreeTextAnswer>
  <Constraints>
    <AnswerFormatRegex regex="^[12][0-9]{3}-[01]?\d-[0-3]?\d$"
     errorText="You must enter a date with the format yyyy-mm-dd."/>
  </Constraints>
</FreeTextAnswer>
```

If you want an answer that contains "regex" and variations including RegEx, REGex, and RegExes, use the following:

```
<FreeTextAnswer>
  <Constraints>
    <AnswerFormatRegex regex="regex" flags="i"
     errorText="You must enter 'regex'."/>
  </Constraints>
</FreeTextAnswer>
```

SelectionAnswer

A `SelectionAnswer` describes a multiple-choice question. Depending on the element defined, the Worker might be able to select zero, one, or multiple items from a set list as the answer to the question.

A `SelectionAnswer` element includes the elements described in the following table:

Name	Description	Required
`MinSelectionCount`	Specifies the minimum number of selections allowed for a valid answer. This value can range from 0 to the number of selections. Type: non-negative Integer Default: 1	No
`MaxSelectionCount`	Specifies the maximum number of selections allowed for a valid answer. This value can range from 1 to the number of selections. Type: positive Integer Default: 1	No
`StyleSuggestion`	Specifies what style of multiple-choice form field to use when displaying the question to the Worker. The field might not use the suggested style, depending on the device the Worker is using to see the form. Type: String Default: None Valid Values:	No

Name	Description	Required
	• Can be used if `MaxSelectionCount` is **1**, because it restricts the user to selecting either zero or one item from the list • Allows multiple selections, but can be restricted by using the `MaxSelectionCount` element • Allows multiple selections, but can be restricted by using the `MaxSelectionCount` element • Can be used if `MaxSelectionCount` is **1**, because it restricts the user to selecting either zero or one item from the list • Allows multiple selections, but can be restricted by using the `MaxSelectionCount` element • Allows multiple selections, but can be restricted by using the `MaxSelectionCount` element	
`Selections`	Specifies the answer selections. Type: `Selections` structure Default: None Child elements: • Specifies an answer selection. This element is described fully in the next table. • An optional text field to display below the selection list that allows the Worker to enter an alternate answer that does not appear in the list of selections. The contents of this element are similar to `FreeTextAnswer`. **Note** A Qualification test that you want to grade automatically using an answer key cannot have an `OtherSelection` field for a multiple choice question. An answer key can only match multiple-choice questions and cannot match free-text fields.	Yes

The `Selections` element lists the selection options available. It contains one or more `Selection` elements, one for each possible answer in the set. The `Selection` element includes the elements described in the following table:

Name	Description	Required
`SelectionIdentifier`	A unique alphanumeric string that is in the answer data if this selection is chosen. Type: String Default: None	Yes
One of the following elements:		Yes

Name	Description	Required
`Text`	Contains the content of the selected item. Type: String Default: None	
`FormattedContent`	A block of text formatted using XHTML tags that contains the content of the selected item. For more information about this format, see Formatted Content: XHTML (p. 206). Type: String Default: None	
`Binary`	Contains the content of the selected item. Type: `Binary` Default: None	

The following example shows a `SelectionAnswer` element that specifies a question with four radiobuttons.

```
<SelectionAnswer>
  <StyleSuggestion>radiobutton</StyleSuggestion>
  <Selections>
    <Selection>
      <SelectionIdentifier>C1</SelectionIdentifier>
      <Text>C1 (northeast)</Text>
    </Selection>
    <Selection>
      <SelectionIdentifier>C2</SelectionIdentifier>
      <Text>C2 (east)</Text>
    </Selection>
    <Selection>
      <SelectionIdentifier>A3</SelectionIdentifier>
      <Text>A3 (southwest)</Text>
    </Selection>
    <Selection>
      <SelectionIdentifier>C3</SelectionIdentifier>
      <Text>C3 (southeast)</Text>
    </Selection>
  </Selections>
</SelectionAnswer>
```

FileUploadAnswer

A `FileUploadAnswer` prompts the Worker to upload a file as the answer to the question. When the Worker uploads the file, Amazon Mechanical Turk stores the file separately from the answer data. Once the HIT is submitted, your application can call the `GetFileUploadURL` operation to get a temporary URL it can use to download the file.

The `FileUploadAnswer` specification contains two required elements, a `MinFileSizeInBytes` and a `MaxFileSizeInBytes`, that specify the minimum and maximum allowed file sizes respectively. If the Worker uploads a file whose size in bytes is outside of this range, the answer is rejected, and the Worker

must upload a different file to complete the HIT. You can specify a maximum size up to 2000000000 (2 billion) bytes.

Note
A `FileUploadAnswer` element can only be used with HITs. It cannot be used with Qualification tests.

The following example demonstrates a `FileUploadAnswer` element that specifies a file with a minimum of 1000 bytes and a maximum of 3000000 bytes.

```
<FileUploadAnswer>
  <MaxFileSizeInBytes>3000000</MaxFileSizeInBytes>
  <MinFileSizeInBytes>1000</MinFileSizeInBytes>
</FileUploadAnswer>
```

Example

The following is an example of a complete `QuestionForm` data structure. Remember that to pass this structure in as a value of a parameter to an operation, XML characters must be escaped as character entities. (See Using XML Parameter Values (p. 191) for more information.)

```
<QuestionForm xmlns="[the QuestionForm schema URL]">
  <Overview>
    <Title>Game 01523, "X" to play</Title>
    <Text>
      You are helping to decide the next move in a game of Tic-Tac-Toe.  The board looks
 like this:
    </Text>
    <Binary>
      <MimeType>
        <Type>image</Type>
        <SubType>gif</SubType>
      </MimeType>
      <DataURL>http://tictactoe.amazon.com/game/01523/board.gif</DataURL>
      <AltText>The game board, with "X" to move.</AltText>
    </Binary>
    <Text>
      Player "X" has the next move.
    </Text>
  </Overview>
  <Question>
    <QuestionIdentifier>nextmove</QuestionIdentifier>
    <DisplayName>The Next Move</DisplayName>
    <IsRequired>true</IsRequired>
    <QuestionContent>
      <Text>
        What are the coordinates of the best move for player "X" in this game?
      </Text>
    </QuestionContent>
    <AnswerSpecification>
      <FreeTextAnswer>
        <Constraints>
          <Length minLength="2" maxLength="2" />
        </Constraints>
        <DefaultText>C1</DefaultText>
      </FreeTextAnswer>
    </AnswerSpecification>
  </Question>
  <Question>
    <QuestionIdentifier>likelytowin</QuestionIdentifier>
    <DisplayName>The Next Move</DisplayName>
    <IsRequired>true</IsRequired>
```

```
    <QuestionContent>
      <Text>
        How likely is it that player "X" will win this game?
      </Text>
    </QuestionContent>
    <AnswerSpecification>
      <SelectionAnswer>
        <StyleSuggestion>radiobutton</StyleSuggestion>
        <Selections>
          <Selection>
            <SelectionIdentifier>notlikely</SelectionIdentifier>
            <Text>Not likely</Text>
          </Selection>
          <Selection>
            <SelectionIdentifier>unsure</SelectionIdentifier>
            <Text>It could go either way</Text>
          </Selection>
          <Selection>
            <SelectionIdentifier>likely</SelectionIdentifier>
            <Text>Likely</Text>
          </Selection>
        </Selections>
      </SelectionAnswer>
    </AnswerSpecification>
  </Question>
</QuestionForm>
```

Formatted Content: XHTML

Topics

When you create a HIT or a Qualification test, you can include various kinds of content to be displayed to the Worker on the Amazon Mechanical Turk web site, such as text (titles, paragraphs, lists), media (pictures, audio, video) and browser applets (Java or Flash).

You can also include blocks of formatted content. Formatted content lets you include XHTML tags directly in your instructions and your questions for detailed control over the appearance and layout of your data.

You include a block of formatted content by specifying a `FormattedContent` element in the appropriate place in your QuestionForm data structure (p. 191). You can specify any number of `FormattedContent` elements in content, and you can mix them with other kinds of content.

The following example uses other content types (`Title`, `Text`) along with `FormattedContent` to include a table in a HIT:

```
<Text>
  This HIT asks you some questions about a game of Tic-Tac-Toe
  currently in progress.  Your answers will help decide the next move.
</Text>
<Title>The Current Board</Title>
<Text>
  The following table shows the board as it currently stands.
</Text>
<FormattedContent><![CDATA[
```

```
<table border="1">
  <tr>
    <td></td>
    <td align="center">1</td>
    <td align="center">2</td>
    <td align="center">3</td>
  </tr>
  <tr>
    <td align="right">A</td>
    <td align="center"><b>X</b></td>
    <td align="center"> </td>
    <td align="center"><b>O</b></td>
  </tr>
  <tr>
    <td align="right">B</td>
    <td align="center"> </td>
    <td align="center"><b>O</b></td>
    <td align="center"> </td>
  </tr>
  <tr>
    <td align="right">C</td>
    <td align="center"> </td>
    <td align="center"> </td>
    <td align="center"><b>X</b></td>
  </tr>
  <tr>
    <td align="center" colspan="4">It is <b>X</b>'s turn.</td>
  </tr>
</table>
]]></FormattedContent>
```

For more information about describing the contents of a HIT or Qualification test, see the QuestionForm data structure (p. 191).

Using Formatted Content

As you can see in the example above, formatted content is specified in an XML CDATA block, inside a `FormattedContent` element. The CDATA block contains the text and XHTML markup to display in the Worker's browser.

Only a subset of the XHTML standard is supported. For a complete list of supported XHTML elements and attributes, see the table below. In particular, JavaScript, element IDs, `class` and `style` attributes, and `<div>` and `<span>` elements are not allowed.

XML comments (`<!-- ... -->`) are not allowed in formatted content blocks.

Every XHTML tag in the CDATA block must be closed before the end of the block. For example, if you start an XHTML paragraph with a `<p>` tag, you must end it with a `</p>` tag within the same `FormattedContent` block.

> **Note**
> The tag closure requirement means you cannot open an XHTML tag in one `FormattedContent` block and close it in another. There is no way to "wrap" other kinds of question form content in XHTML. `FormattedContent` blocks must be self-contained.

XHTML tags must be nested properly. When tags are used inside other tags, the inner-most tags must be closed before outer tags are closed. For example, to specify that some text should appear in bold italics, you would use the `<b>` and `<i>` tags as follows:

```
<b><i>This text appears bold italic.</i></b>
```

But the following would not be valid, because the closing `</b>` tag appears before the closing `</i>` tag:

```
<b><i>These tags don't nest properly!</b></i>
```

Finally, formatted content must meet other requirements to validate against the XHTML schema. For instance, tag names and attribute names must be all lowercase letters, and attribute values must be surrounded by quotes.

For details on how Amazon Mechanical Turk validates XHTML formatted content blocks, see "How XHTML Formatted Content Is Validated," below.

Supported XHTML Tags

`FormattedContent` supports a limited subset of the XHTML 1.0 ("transitional") standard. The complete list of supported tags and attributes appears in the table below. Notable differences with the standard include:

- JavaScript is not allowed. The `<script>` tag is not supported, and anchors (`<a>`) and images (`<img>`) cannot use `javascript:` targets in URLs.
- CSS is not allowed. The `<style>` tag is not supported, and the `class` and `style` attributes are not supported. The `id` attribute is also not supported.
- XML comments (`<!-- ... -->`) are not supported.
- URL methods in anchor targets and image locations are limited to the following: `http:// https:// ftp:// news:// nntp:// mailto:// gopher:// telnet://`

Other things to note with regards to supported tags and attributes:

- In addition to the attributes listed, the `title` attribute is supported for all tags, and the `dir` and `lang` attributes are supported for all tags except `<br>`.
- The `alt` attribute is required for `<area>` and `<img>` tags.
- `<img>` tags also require a `src` attribute.
- `<map>` tags require a `name` attribute.

The following table lists the supported tags and attributes:

Tag	Attributes
a	accesskey charset coords href hreflang name rel rev shape tabindex target type
area	alt coords href nohref shape target
b	
big	
blockquote	cite
br	
center	
cite	
code	
col	align char charoff span valign width

Tag	Attributes
colgroup	align char charoff span valign width
dd	
del	cite datetime
dl	
em	
font	color face size
h1	align
h2	align
h3	align
h4	align
h5	align
h6	align
hr	align noshade size width
i	
img	align alt border height hspace ismap longdesc src usemap vspace width
ins	cite datetime
li	type value
map	name
ol	compact start type
p	align
pre	width
q	cite
small	
strong	
sub	
sup	
table	align bgcolor border cellpadding cellspacing frame rules summary width
tbody	align char charoff valign
td	abbr align axis bgcolor char charoff colspan headers height nowrap rowspan scope valign width

Tag	Attributes
tfoot	align char charoff valign
th	abbr align axis bgcolor char charoff colspan headers height nowrap rowspan scope valign width
thead	align char charoff valign
tr	align bgcolor char charoff valign
u	
ul	compact type

How XHTML Formatted Content Is Validated

When you create a HIT or a Qualification test whose content uses `FormattedContent`, Amazon Mechanical Turk attempts to validate the formatted content blocks against a schema. If the formatted content does not validate against the schema, the operation call will fail and return an error.

To validate the formatted content, Amazon Mechanical Turk takes the contents of the `FormattedContent` element (the text and markup inside the CDATA), then constructs an XML document with an appropriate XML header, `<FormattedContent>` as the root element, and the text and markup as the element's contents (without the CDATA). This document is then validated against a schema.

For example, consider the following `FormattedContent` block:

```
...
<FormattedContent><![CDATA[
  I absolutely <i>love</i> chocolate ice cream!
]]></FormattedContent>
...
```

To validate this block, Amazon Mechanical Turk produces the following XML document:

```
<?xml version="1.0"?>
<FormattedContent xmlns="http://www.w3.org/1999/xhtml">
  I absolutely <i>love</i> chocolate ice cream!
</FormattedContent>
```

The schema used for validation is called `FormattedContentXHTMLSubset.xsd`. For information on how to download this schema, see WSDL and Schema Locations (p. 2).

You do not need to specify the namespace of the XHTML tags in your formatted content. This is assumed automatically during validation.

QuestionFormAnswers

Topics

Description

The `QuestionFormAnswers` data format describes answers submitted by a Worker for a HIT, or for a Qualification test.

A `QuestionFormAnswers` data structure is a string value that consists of XML data. The XML data must conform to the QuestionForm schema. See WSDL and Schema Locations (p. 2) for the location of this schema. For more information about using XML data as parameter or return value, see Using XML Parameter Values (p. 191).

> **Note**
> Answer data is *not* guaranteed by the Amazon Mechanical Turk Service to conform to the answer specifications described in a QuestionForm. MTS only guarantees that answer data returned by the service will conform to the `QuestionFormAnswers` schema. Your application should check that the answer data sufficiently answers the question.

The `QuestionFormAnswers` data structure is used as a response element for the following operations:

- `GetAssignmentsForHIT`
- `GetQualificationRequests`

The `QuestionFormAnswers` data structure is a value in an Assignment data structure (p. 140), and a value in a QualificationRequest data structure (p. 161).

All elements in a `QuestionFormAnswers` belong to a namespace whose name is identical to the URL of the `QuestionFormAnswers` schema document for the version of the API you are using.

The Structure of Answers

A `QuestionFormAnswers` element contains an `Answer` element for each question in the HIT or Qualification test for which the Worker provided an answer. Each `Answer` contains a `QuestionIdentifier` element whose value corresponds to the `QuestionIdentifier` of a `Question` in the QuestionForm. See the QuestionForm data structure (p. 191) for more information about questions and answer specifications.

If the question expects a free-text answer, the `Answer` element contains a `FreeText` element. This element contains the Worker's answer.

If the question expects a multiple-choice answer, the `Answer` element contains a `SelectionIdentifier` element for each option the Worker selected. If the Worker did not make any selections, the `Answer` will contain zero `SelectionIdentifier` elements. The identifier corresponds to the `SelectionIdentifier` for the selection provided in the answer specification for the question.

If the multiple-choice question includes an `OtherSelection` field, and the Worker enters data into this field, that data appears in the `Answer` in an `OtherSelectionText` element. If the Worker both selects an option from the list and provides text in this field, both values will be present in the answer.

If the question expects an uploaded file as an answer, the `Answer` element contains an `UploadedFileSizeInBytes` element, and an `UploadedFileKey` element. `UploadedFileSizeInBytes` indicates the size of the file the Worker uploaded. `UploadedFileKey` is a unique identifier for the file, unique with respect to other files that Workers may have uploaded. To retrieve an uploaded file, your application calls the `GetFileUploadURL` operation, which returns a temporary URL your application can use to download the file. See the GetFileUploadURL operation (p. 62) for more information on retrieving uploaded files.

Answer data will always conform to the answer specification provided in the HIT question, or in the Qualification test question.

Example

The following is an example of a complete `QuestionFormAnswers` data structure. Remember that this value will be returned as a single return value, XML escaped in the response.

```
<QuestionFormAnswers xmlns="[the QuestionFormAnswers schema URL]">
  <Answer>
    <QuestionIdentifier>nextmove</QuestionIdentifier>
    <FreeText>C3</FreeText>
  </Answer>
  <Answer>
    <QuestionIdentifier>likelytowin</QuestionIdentifier>
    <SelectionIdentifier>notlikely</SelectionIdentifier>
  </Answer>
</QuestionFormAnswers>
```

AnswerKey

Topics

Description

The AnswerKey data structure specifies answers for a Qualification test, and a mechanism to use to calculate a score from the key and a Worker's answers.

An AnswerKey data structure is a string value that consists of XML data. The XML data must conform to the AnswerKey schema. See WSDL and Schema Locations (p. 2) for the location of this schema. For more information about using XML data as parameter or return value, see Using XML Parameter Values (p. 191).

The AnswerKey data structure is used as a parameter for the following operations:

- `CreateQualificationType`

The AnswerKey data structure is used as a return value for the following operations:

- `GetQualificationType`

The AnswerKey data structure is a value in a Qualification type data structure (p. 171).

All elements in a `AnswerKey` belong to a namespace whose name is identical to the URL of the `AnswerKey` schema document for the version of the API you are using.

The Structure of an Answer Key

An answer key is contained in a `AnswerKey` element. This element contains a `Question` element for each question in the Qualification test, and an optional `QualificationValueMapping` element that describes how to calculate the Qualification value from the answer key and the Worker's answers.

Question

A `Question` element contains a `QuestionIdentifier` element, which identifies the question for this answer. This value corresponds to a `QuestionIdentifier` in the QuestionForm.

A `Question` element has one or more `AnswerOption` elements, one for each combination of selections in the multiple-choice question that affects the Worker's test score.

Each `AnswerOption` contains one or more `SelectionIdentifier` elements that correspond to identifiers for the selections in the QuestionForm. It also contains an `AnswerScore` element, a number that is added to the Worker's test score if the Worker's answer matches this option. The Worker must select all of the selections specified by the `SelectionIdentifier` elements, and no others, to earn the score.

Tip
An `AnswerScore` for an `AnswerOption` may be negative.

The `Question` may have an optional `DefaultScore`, a number that is added to the Worker's test score if none of the answer options exactly match the Worker's answer for the question. `DefaultScore` is optional, and defaults to 0.

```
<AnswerOption>
  <SelectionIdentifier>apples</SelectionIdentifier>
  <AnswerScore>10</AnswerScore>
</AnswerOption>
```

QualificationValueMapping

The `Question` may have an optional `QualificationValueMapping` element that describes how to calculate the Worker's overall score from the scores of the Worker's answers. It contains either a `PercentageMapping` element, a `ScaleMapping` element, or a `RangeMapping` element.

If no `QualificationValueMapping` is specified, the sum of the scores of the answers is used as the Qualification value.

```
<QualificationValueMapping>
  ...
</QualificationValueMapping>
```

A `PercentageMapping` specifies a maximum score for the test, as a `MaximumSummedScore` element. The Qualification value is calculated as the sum of the scores of the selected answers, divided by the maximum, multiplied by 100 and rounded to the nearest integer to produce a percentage.

```
  ...
  <PercentageMapping>
    <MaximumSummedScore>15</MaximumSummedScore>
  </PercentageMapping>
```

A `ScaleMapping` specifies a multiplier, as a decimal value in a `SummedScoreMultiplier` element. The Qualification value is calculated as the sum of the scores of the selected answers, multiplied by the multiplier.

```
...
<ScaleMapping>
  <SummedScoreMultiplier>3</SummedScoreMultiplier>
</ScaleMapping>
```

A `RangeMapping` assigns specific Qualification values to ranges of total test scores. It contains one or more `SummedScoreRange` elements, each of which specify an `InclusiveLowerBound` element, an `InclusiveUpperBound` element, and a `QualificationValue` that becomes the Qualification value if the sum of the scores of the selected answers falls within the specified range. Finally, the `RangeMapping` includes a single `OutOfRangeQualificationValue`, which specifies the Qualification value if the sum of the scores of the selected answers do not fall within a specified range.

Note
Ranges cannot overlap. If ranges overlap, the behavior is undefined.

```
...
<RangeMapping>
  <SummedScoreRange>
    <InclusiveLowerBound>5</InclusiveLowerBound>
    <InclusiveUpperBound>7</InclusiveUpperBound>
    <QualificationValue>5</QualificationValue>
  </SummedScoreRange>
  <SummedScoreRange>
    <InclusiveLowerBound>8</InclusiveLowerBound>
    <InclusiveUpperBound>10</InclusiveUpperBound>
    <QualificationValue>10</QualificationValue>
  </SummedScoreRange>
  <OutOfRangeQualificationValue>0</OutOfRangeQualificationValue>
</RangeMapping>
```

Example

The following is an example of a complete `AnswerKey` data structure. Remember that to pass this structure in as a parameter to an operation, XML characters must be escaped as character entities. For more information, see Using XML Parameter Values (p. 191).

```
<AnswerKey xmlns="[the AnswerKey schema URL]">
  <Question>
    <QuestionIdentifier>nextmove</QuestionIdentifier>
    <AnswerOption>
      <SelectionIdentifier>D</SelectionIdentifier>
      <AnswerScore>5</AnswerScore>
    </AnswerOption>
  </Question>
  <Question>
    <QuestionIdentifier>favoritefruit</QuestionIdentifier>
    <AnswerOption>
      <SelectionIdentifier>apples</SelectionIdentifier>
      <AnswerScore>10</AnswerScore>
    </AnswerOption>
  </Question>
  <QualificationValueMapping>
    <PercentageMapping>
      <MaximumSummedScore>15</MaximumSummedScore>
    </PercentageMapping>
  </QualificationValueMapping>
```

```
</AnswerKey>
```

ExternalQuestion

Topics

Description

Instead of providing a QuestionForm data structure (p. 191) that tells Amazon Mechanical Turk how to display your questions and collect answers, you can host the questions on your own website using an "external" question.

A HIT with an external question displays a web page from your website in a frame in the Worker's web browser. Your web page displays a form for the Worker to fill out and submit. The Worker submits results using your form, and your form submits the results back to Amazon Mechanical Turk. Using your website to display the form gives your website control over how the question appears and how answers are collected.

To use an external question with a HIT, you provide an `ExternalQuestion` data structure as the value of the `Question` parameter when calling the CreateHIT (p. 21) operation. As with the `QuestionForm` data structure, an `ExternalQuestion` is a string value that consists of XML data. This data must conform to the `ExternalQuestion` schema. See WSDL and Schema Locations (p. 2) for the location of this schema. For more information about using XML data as a parameter or return value, see Using XML Parameter Values (p. 191).

> **Note**
> You can only use an external question as the question of a HIT. You cannot use an external question with a Qualification test.

The `ExternalQuestion` data structure is a value in a HIT (p. 144) data structure.

All elements in a `ExternalQuestion` belong to a namespace whose name is identical to the URL of the `ExternalQuestion` schema document for the version of the API you are using.

The ExternalQuestion Data Structure

The `ExternalQuestion` data structure has a root element of **`ExternalQuestion`**.

The **`ExternalQuestion`** element contains the following elements:

Name	Description	Required
`ExternalURL`	The URL of your web form, to be displayed in a frame in the Worker's web browser. This URL must use the HTTPS protocol. Type: URL Default: None	Yes

Name	Description	Required
	Amazon Mechanical Turk appends the following parameters to this URL: `assignmentId`, `hitId`, `turkSubmitTo`, and `workerId`. For more information about these appended parameters, see the sections following this table.	
`FrameHeight`	The height of the frame, in pixels. Type: Integer Default: None	Yes

Example

The following is an example of a complete `ExternalQuestion` data structure. Remember that to pass this structure in as the value of a parameter to an operation, XML characters must be escaped as character entities. For more information about escaping XML characters, see Using XML Parameter Values (p. 191). For information on the ExternalQuestion schema URL, see WSDL and Schema Locations (p. 2).

```
<ExternalQuestion xmlns="[the ExternalQuestion schema URL]">
  <ExternalURL>https://tictactoe.amazon.com/gamesurvey.cgi?gameid=01523</ExternalURL>
  <FrameHeight>400</FrameHeight>
</ExternalQuestion>
```

The External Form

When a Worker attempts to complete a HIT with an external question, the external website is loaded into a frame in the middle of the screen. The web page at that URL should display a form for the Worker to fill out, and all the information the Worker will need to complete the HIT.

The Frame's URL and Parameters

The URL used for the frame is the `ExternalURL` of the question with the following parameters appended: `assignmentId`, `hitId`, `turkSubmitTo`, and `workerId`. These parameters are appended CGI-style: The full URL has a question mark (`?`) before the first parameter, and an ampersand (`&`) between each parameter, with each parameter consisting of a name, an equal sign (`=`), and a value. Other parameters already present in this style in `ExternalURL` are preserved, so the final URL will only have one question mark, and all parameters will be separated by ampersands (`&`).

Note
The URL you use for the `ExternalURL` must use the HTTPS protocol.

For example, consider an `ExternalURL` of:

```
https://tictactoe.amazon.com/gamesurvey.cgi?gameid=01523
```

With this `ExternalURL`, the full URL used for the page in the frame could be as follows:

```
https://tictactoe.amazon.com/gamesurvey.cgi?gameid=01523
&assignmentId=123RVWYBAZW00EXAMPLE456RVWYBAZW00EXAMPLE
&hitId=123RVWYBAZW00EXAMPLE
&turkSubmitTo=https://www.mturk.com/
```

```
&workerId=AZ3456EXAMPLE
```

Preview Mode

Your external question will be displayed when a Worker previews the HIT on the Amazon Mechanical Turk website, before the Worker has clicked the "Accept HIT" button. When the HIT is being previewed, the URL will have a special value for the `assignmentId`: `ASSIGNMENT_ID_NOT_AVAILABLE`.

When a Worker previews a HIT, your web page should show her everything she will need to do to complete the HIT, so she can decide whether or not to accept it. The easiest way to do this is to simply display the form as it would appear when the HIT is accepted. However, you may want to take precautions to prevent a Worker from accidentally filling out or submitting your form prior to accepting the HIT.

You can use JavaScript or server-side logic to check the `assignmentId` parameter, and change the display of the form if the HIT is being previewed (`assignmentId=ASSIGNMENT_ID_NOT_AVAILABLE`).

If a Worker submits your form before accepting the HIT, and your form attempts to post the data back to Amazon Mechanical Turk, Amazon Mechanical Turk will display an error message to the Worker, and the results will not be accepted.

The Form Action

The form on the external website must post the result data back to Amazon Mechanical Turk using the following URL:

```
https://www.mturk.com/mturk/externalSubmit
```

Or, if you are using the Amazon Mechanical Turk sandbox, you should post the result data back to Mechanical Turk using the following sandbox URL:

```
https://workersandbox.mturk.com/mturk/externalSubmit
```

The form must include the `assignmentId` field that was appended to the URL used to access your form. It should be submitted along with the other form fields submitted by your form, with a name of `assignmentId` and the same value as was passed to the form. Be sure to spell the field name as it appears here, with the same letters uppercase and lowercase.

> **Note**
> The field names `assignmentId`, `hitId`, `turkSubmitTo`, and `workerId` are reserved for special purposes. Your form only needs to submit the `assignmentId` field. Any data submitted with a field name of "`hitId`" will be ignored, and will not appear in the results data for the HIT.

The form must submit data to that URL using the "POST" method. The data the form submits should be name-value pairs in the CGI-style:

- Each field appears as the name, an equal sign, and the value. For example: `favoriteColor=blue`
- Data that appears in the posted URL is preceded by a question mark (`?`), and is delimited by ampersands (`&`). For example:

  ```
  https://www.mturk.com/mturk/externalSubmit?favoriteColor=blue&favoriteNumber=7&...
  ```

- Data that appears in the HTTP message body (using the "POST" method) has one data pair per line. For example:

  ```
  favoriteColor=blue
  favoriteNumber=7
  ```

```
...
```

The easiest way to post the data in the CGI-style is to use an HTML form on the web page, with the `externalSubmit` URL as the "action," and "POST" as the "method."

The Answer Data

When the Worker submits your form, the form sends the field data to Amazon Mechanical Turk using the `externalSubmit` URL, and Amazon Mechanical Turk records the field data as the results of the Assignment.

When you retrieve the results using the `GetAssignmentsForHIT` operation (p. 52), the field data submitted by your form will appear in the `Answer` of the Assignment (p. 140) as if each field were a free-text answer. The `QuestionIdentifier` element of the answer will be the name of the field, and the `FreeText` element will contain the value.

See the QuestionFormAnswers data format (p. 211) for more information about the format of answer data.

Guidelines For Using External Questions

External questions give your application a great deal of power over how Workers submit results for your HITs. To ensure you get good results for your HITs, you should make sure your web server and web pages can provide your Workers with a quality experience.

Because external questions depend on your web server for rendering the question form, both while Workers are previewing HITs and while Workers are completing HITs, your server will need to be engineered for high availability. The Amazon Mechanical Turk website gets heavy traffic, so your web server will need to be able to respond quickly and correctly when receiving many requests in a short period of time.

> **Tip**
> Amazon S3 offers high availability hosting of data, accessible via public URLs. You can host your external questions as web pages in Amazon S3, and not have to run your own high availability web server.

Your website can do many things inside the frame, but eventually it must cause the Worker's browser to load the "externalSubmit" URL in the frame with the results in POST data. The easiest way to do this is with an HTML form whose fields contain the HIT results, with a submit button that the Worker will click. If an external HIT prevents the Worker from submitting results back to Amazon Mechanical Turk using the "externalSubmit" mechanism, the Worker may not be able to claim rewards or continue doing work without restarting their session. Amazon Mechanical Turk reserves the right to remove any external HITs that are not functioning properly.

> **Note**
> Your HIT will be rendered inside an IFRAME that has certain limitations. The IFRAME operates in HTML5 "sandbox" mode that has extra restrictions on the content that can appear in the frame. This limits your ability to execute certain code and to use technologies such as Adobe Flash. To ensure your HITs work as expected, we recommend you test them first in the Requester Sandbox.

Finally, please remember that external questions must meet the Amazon Mechanical Turk Participation Agreement, and Amazon Mechanical Turk's standards for appropriate content. Specifically, the Participation Agreement expressly prohibits the use of Amazon Mechanical Turk for advertising or solicitation. If your website typically displays advertising to visitors, please make sure those advertisements do not appear in your external questions. Amazon Mechanical Turk reserves the right to remove HITs with inappropriate content.

HTMLQuestion

Topics

Description

The `HTMLQuestion` data structure defines one or more questions for a HIT using HTML. The `HTMLQuestion` data structure is similar to both the `QuestionForm` and `ExternalQuestion` data structures.

The `QuestionForm` data structure defines, using a special XML language, how Amazon Mechanical Turk displays HIT questions and collects the answers. The `ExternalQuestion` data structure defines, using HTML, questions you host on your own "external" website. If you want to define your questions using HTML forms without having to host a website, you can use the `HTMLQuestion` data structure.

A `HTMLQuestion` HIT is like a cross between a `QuestionForm` HIT and an `ExternalQuestion` HIT, for instance:

- Like a `QuestionForm` HIT, you do not need to run a website or run any other infrastructure to have your HIT display on Mechanical Turk. You define your question when you call CreateHIT and then collect worker answers later, after they have been submitted.
- Like an `ExternalQuestion` HIT, you can define your HIT in HTML. Your HTML code must contain a form for the Worker to fill out and submit, which is displayed in a frame in the Worker's web browser. The Worker submits results using your form, and your form submits the results back to Amazon Mechanical Turk. Worker answers are processed by Mechanical Turk in the same way as `ExternalQuestion` HITs. If you choose, you can collect or process the results before submitting to Mechanical Turk.

The worker interaction and presentation options available for `HTMLQuestion` are similar to `ExternalQuestion`. HTMLQuestions differ from ExternalQuestions primarily in how they are created.

As with the other question data structures, an `HTMLQuestion` is a string value that consists of XML data. This data must conform to the `HTMLQuestion` schema. See WSDL and Schema Locations (p. 2) for the location of this schema. For more information about using XML data as a parameter or return value, see Using XML Parameter Values (p. 191).

> **Note**
> You can only use an `HTMLQuestion` as the question of a HIT. You cannot use an `HTMLQuestion` with a Qualification test.

The `HTMLQuestion` data structure is used as a parameter value for the following operation:

- `CreateHIT`

The `HTMLQuestion` data structure is a value in a HIT (p. 144) data structure.

All elements in an `HTMLQuestion` belong to a namespace whose name is identical to the URL of the `HTMLQuestion` schema document for the version of the API you are using.

The HTMLQuestion Data Structure

The `HTMLQuestion` data structure has a root element of `HTMLQuestion`.

The `HTMLQuestion` element contains the following elements:

Name	Description	Required
`HTMLContent`	The HTML code of your web form, to be displayed in a frame in the Worker's web browser. The HTML must validate against the HTML5 specification. HTML5 is backwards-compatible with a variety of recent HTML document specifications. For more information, see http://www.w3.org/TR/html5-diff/. For help in ensuring that your HTML validates, see http://validator.w3.org. Type: String Default: None Amazon Mechanical Turk appends the following parameters to this URL: `assignmentId`, `hitId`, `turkSubmitTo`, and `workerId`. For more information about these appended parameters, see the sections following this table.	Yes
`FrameHeight`	The height of the frame, in pixels. Type: Integer Default: None	Yes

Example

The following is an example of a complete `HTMLQuestion` data structure. Remember that to pass this structure in as the value of a parameter to an operation, XML characters must be escaped as character entities. For more information, see Using XML Parameter Values (p. 191).

```
<HTMLQuestion xmlns="[the HTMLQuestion schema URL]">
  <HTMLContent><![CDATA[
<!DOCTYPE html>
<html>
 <head>
  <meta http-equiv='Content-Type' content='text/html; charset=UTF-8'/>
  <script type='text/javascript' src='https://s3.amazonaws.com/mturk-public/
externalHIT_v1.js'></script>
 </head>
 <body>
  <form name='mturk_form' method='post' id='mturk_form' action='https://www.mturk.com/
mturk/externalSubmit'>
  <input type='hidden' value='' name='assignmentId' id='assignmentId'/>
  <h1>What's up?</h1>
  <p><textarea name='comment' cols='80' rows='3'></textarea></p>
  <p><input type='submit' id='submitButton' value='Submit' /></p></form>
```

```
  <script language='Javascript'>turkSetAssignmentID();</script>
 </body>
</html>
]]>
  </HTMLContent>
  <FrameHeight>450</FrameHeight>
</HTMLQuestion>
```

Preview Mode

The question defined by `HTMLQuestion` displays when a Worker previews the HIT on the Amazon Mechanical Turk website, before the Worker clicks the **Accept HIT** button. When the HIT is being previewed, the URL has a special value for the `assignmentId`: `ASSIGNMENT_ID_NOT_AVAILABLE`. This is the same mechanism used for `ExternalQuestion` HITs.

When a Worker previews a HIT, your HTML should show the Worker everything they will need to do to complete the HIT, so they can decide whether or not to accept it. The easiest way to do this is to simply display the form as it would appear when the HIT is accepted. However, you may want to take precautions to prevent a Worker from accidentally filling out or submitting your form prior to accepting the HIT.

You can use JavaScript to check the `assignmentId` parameter, and change the display of the form if the HIT is being previewed (`assignmentId=ASSIGNMENT_ID_NOT_AVAILABLE`).

The Form Action

For information about form actions for `HTMLQuestion`, see " The Form Action" in ExternalQuestion (p. 216).

The Answer Data

For information about answer data for `HTMLQuestion`, see " The Answer Data" in ExternalQuestion (p. 216).

Guidelines For Using HTML Questions

Tip
Your HTML code can do many things inside the browser frame, but eventually it must cause the Worker's browser to load the "externalSubmit" URL in the frame with the results in POST data. The easiest way to do this is with an HTML form whose fields contain the HIT results, with a submit button that the Worker clicks. If a `HTMLQuestion` HIT prevents the Worker from submitting results back to Amazon Mechanical Turk using the "externalSubmit" mechanism, the Worker may not be able to claim rewards or continue doing work without restarting their session. Amazon Mechanical Turk reserves the right to remove any `HTMLQuestion` HITs that are not functioning properly.

Note
Your HIT will be rendered inside an IFRAME that has certain limitations. The IFRAME operates in HTML5 “sandbox” mode that has extra restrictions on the content that can appear in the frame. This limits your ability to execute certain code and to use technologies such as Adobe Flash. To ensure your HITs work as expected, we recommend you test them first in the Requester Sandbox.

Tip
All `HTMLQuestion` HITs are served from the same domain, regardless of requester. Bear this in mind if you choose to set cookies from JavaScript in your HTML.

HITLayout

Topics

Description

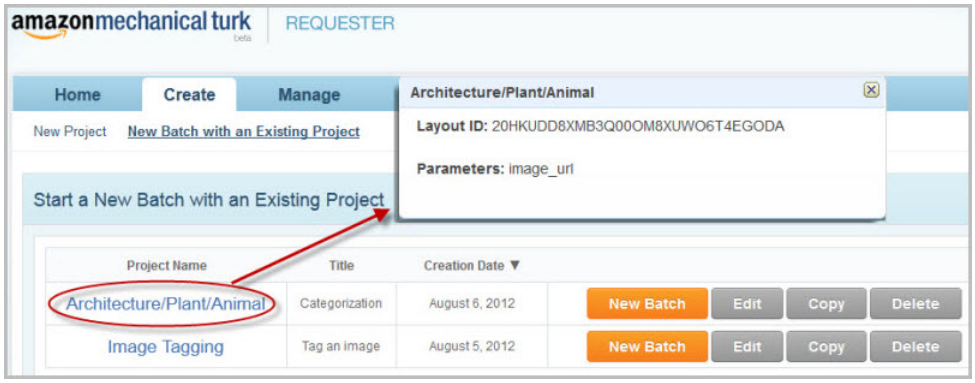
A HITLayout is a reusable Amazon Mechanical Turk project template used to provide Human Intelligence Task (HIT) question data for CreateHIT (p. 21). You can create a HITLayout template by creating a Mechanical Turk project on the Amazon Mechanical Turk Requester website. For more information about creating a project, see How to Create a Project in the Requester UI Guide.

Obtaining a Layout ID

A **Layout ID** is assigned to each Mechanical Turk project you create on the Requester website. You use the **Layout ID** as the value for `HITLayoutId` when calling `CreateHIT` to identify the HITLayout project template to use. Mechanical Turk projects can contain parameter placeholders in the format `${parameter_name}`. The names for the parameter placeholders used in a HITLayout project template are listed as **Parameters** along with the **Layout ID** on the Requester website.

To view the Layout ID and the Parameters used in your HITLayout project template

1. Go to the Amazon Mechanical Turk Requester website. Or for the Requester Sandbox site, go to the Amazon Mechanical Turk Requester Sandbox website.
2. Click **Create**, and then click **New Batch with an Existing Project**.
3. Click the **Project Name** of an existing project to view **Layout ID** and **Parameters**.



Using a HITLayout

You can use the HITLayout form of a HIT by calling `CreateHIT` with a `HITLayoutId` and a list of HITLayoutParameter (p. 150) structures. The project parameter placeholders are replaced with values from the HITLayoutParameter (p. 150) structures when you call `CreateHIT` to create a HIT. You need one structure for each of the parameter values you want substituted. The parameter names that you pass to `CreateHIT` must match the parameter names used in the HITLayout project template created on the Requester website. The parameter values cannot be changed after the HIT has been created.

Note
You can use either the `HITLayoutId` or the `Question` parameter when calling `CreateHIT`, but not both.

Each `CreateHIT` call merges the parameter values from HITLayoutParameter structures into the HITLayout template to generate the HIT question document. You use the same **Layout ID** in `HITLayoutId` to call `CreateHIT` multiple times, with different parameter values supplied each time for the placeholders.

Requesters can use this parameter substitution capability to create a large number of HITs that all share a common design. For example, you can create a HIT question that asks Workers to provide keywords for an image and draw boxes around key image features using a JavaScript library. First, you use the Requester website to create a Mechanical Turk project that uses a parameter placeholder for the image URL. Then you call `CreateHIT` using the same HITLayout template iteratively, using a different image URL value each time. Each call to `CreateHIT` uses the same **Layout ID**, but each call uses a different HITLayoutParameter structure that contains a unique image URL.

Guidelines for Using HITLayouts

- After a HIT is created, the HIT behaves like an HTMLQuestion HIT, which gives you the option to use HTML and JavaScript features in your HIT design, including Asynchronous JavaScript and XML (AJAX) callbacks.
- Parameter substitution allows you to replace a short parameter name with long strings of text. You will receive errors if the resulting document is longer than permitted by the `Question` parameter of `CreateHIT`.
- The HITLayout is used to create an HTMLQuestion document. HITLayoutParameter values with reserved characters or invalid HTML markup may result in an invalid HTMLQuestion document. For more information, see HTMLQuestion (p. 220).

The Notification API

Topics

This section describes how to set up and handle Amazon Mechanical Turk event notification messages. A notification message describes one or more events that happened in regards to a HIT type. For more information, see Elements of a Notification Message (p. 225).

You can configure Amazon Mechanical Turk to notify you whenever certain events occur during the life cycle of a HIT. Mechanical Turk can send you a notification message when a Worker accepts, abandons, returns, or submits an assignment, when a HIT becomes "reviewable", or when a HIT expires, for any HIT of a given HIT type.

Notifications are specified as part of a HIT type. To set up notifications for a HIT type, you call the SetHITTypeNotification (p. 129) operation with a HIT type ID and a notification specification. For more information about HIT types, see Understanding HIT Types.

A notification specification is defined by a Notification (p. 175) data structure, which describes a HIT event notification for the HIT type. The notification specification is passed as the `Notification` parameter when calling SetHITTypeNotification (p. 129).

Amazon Mechanical Turk can send a notification to an email address or to an Amazon Simple Queue Service (Amazon SQS) queue. For more information, see Notification Handling Using Amazon SQS (p. 226).

For more information about setting up and handling notifications, see Creating and Managing Notifications.

You can test your application's ability to receive notifications using SendTestEventNotification (p. 125).

> **Note**
> The latest Amazon Mechanical Turk WSDL includes deprecated notification transport signature protocols for backwards compatibility.

Elements of a Notification Message

Notification messages contain one or more `Event` data structures that describe recent activity for HITs of a HIT type.

The Notification API Version

Similar to how a REST request that is sent to the Amazon Mechanical Turk Requester service must include a `Version` parameter to indicate which version of the service API the client is expecting to use, a notification message must also include a `Version` parameter. This version string is identical to the version that is included in the notification specification for the HIT type.

> **Tip**
> Your application may need to accommodate receiving notification messages of different versions at the same time if you want to upgrade your notification specifications to a new version without missing messages. You can avoid having to accommodate multiple API versions by first disabling the notification specifications that use the old version, upgrading your

application to use the new version, then updating the notification specifications to use the new version and re-enable notifications.

When a new version of the notification API is made available, all existing notification specifications will continue to use the API versions they were using previously. You must update your notification specifications to use a new version of the API.

Events

A notification message describes one or more events that happened in regards to a HIT type. Each event includes:

- the event type (`EventType`), a value corresponding to the `EventType` value in the notification specification data structure (p. 175)
- the time of the event (`EventTime`), as a dateTime in the Coordinated Universal Time time zone, such as **`2005-01-31T23:59:59Z`**
- the HIT type ID for the event (`HITTypeId`)
- the HIT ID for the event (`HITId`)
- the assignment ID for the event, if applicable (`AssignmentId`)

Multiple events may be batched into a single notification message.

Notification Handling Using Amazon SQS

Your application can use the Amazon Simple Queue Service (Amazon SQS) to handle Mechanical Turk notifications. By using Amazon SQS, your notifications are guaranteed to be delivered at least once. For more information about guaranteed delivery of notifications, see Guaranteed Delivery (p. 228). For more information about, see Amazon SQS.

Creating an SQS Queue

You must create an Amazon SQS queue before using the SQS transport type in notification-related calls. Mechanical Turk does not create an Amazon SQS queue for you. An SQS queue can be created through the Amazon SQS API or by using the AWS Console. For more information, see the Amazon SQS documentation.

Configuring an SQS Queue

Your Amazon SQS queue permissions must be configured to allow a Mechanical Turk system account to call the `sqs:SendMessage'` action on your queue. Whether you use the management console UI or the API to configure permissions, consider the following:

- You must add a permission that enables the account principal 'arn:aws:iam::755651556756:user/MTurk-SQS' to call SendMessage on your queue.
- Your SendMessage permission must add an action of 'aws:SecureTransport' set to 'true'.

  Mechanical Turk always uses the secure transport; this ensures that you only enable the level of access that will actually be used.
- You must not provide any additional permissions to this account to ensure that you only allow the access that will actually be used.
- You should consider disallowing all other access to your queue from other accounts.

  This makes it easy for you to be sure that available messages were sent by Mechanical Turk.

If you enable SendMessage for other accounts to this queue, or if you plan to send messages to this queue from your AWS account, you should check the sending identity for every message that you receive from the queue. You can do this by requesting the `SenderId` attribute in your call to ReceiveMessage. This value will be `AIDAIXO4EZE6RHVSXIN4E`. Amazon SQS provides this value as a strong guarantee of the authenticated identity of the sender, so if it matches, you can be sure the message came from Mechanical Turk.

For more information, see the Amazon SQS Developer Guide and Amazon SQS API Reference.

Amazon SQS Policy Document Example

The following example policy document only creates the SendMessage permission for the Mechanical Turk account. You can add additional restrictions, such as setting the `aws:SecureTransport` value. For more information about policy documents, see the Amazon SQS Developer Guide.

```
{
 "Version": "2008-10-17",
 "Id": "arn:aws:sqs:us-east-1:<customer_account_id>:<customer_queue_name>/MTurkOnlyPolicy",
 "Statement": [
  {
   "Sid": "MTurkOnlyPolicy",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::755651556756:user/MTurk-SQS"
   },
   "Action": "SQS:SendMessage",
   "Resource": "arn:aws:sqs:us-east-1:<customer_account_id>:<customer_queue_name>"
  }
 ]
}
```

Configuring Permissions Using the AWS Console

To configure permissions in the AWS Console:

1. Sign in to the AWS Management Console and open the Amazon SQS console at https://console.aws.amazon.com/sqs/.
2. Select your queue, and then select **Permissions**.
3. Click **Edit Policy Document**.
4. Enter a policy document similar to the example.

Configuring Permissions Using the Amazon SQS API

Call the Amazon SQS SetQueueAttributes action with the `Attribute.Name` parameter set to Policy. You can call `SetQueueAttributes` with a policy document similar to the example policy document. Do not use the Amazon SQS `AddPermission` action for configuring permissions on this queue. If you programmatically create a queue and apply a policy document to it, you must ensure the 'Resource' value in the policy document is updated with the correct queue name.

Testing Your Queue

To test your permissions, call the Mechanical Turk SendTestEventNotification (p. 125) operation with a `Transport` of 'SQS' and your queue URL as the Destination.

Guaranteed Delivery

Using Amazon SQS provides a guaranteed at-least-once delivery of each message. Mechanical Turk ensures that it calls SendMessage at least once for each message. SQS then provides guarantees regarding message persistence and message delivery.

Rarely, the same message may show up twice in the queue. This is an attribute of Amazon SQS's nature as a distributed system.

If you take action on your queue that prevents Mechanical Turk from publishing to it, we cannot guarantee delivery of the messages that would have been sent to your queue. For instance, such actions may include:

- Modifying the permissions on your queue in a way that prevents our account from calling SendMessage successfully.
- Deleting or disabling your queue.

SQS Message Ordering

You should expect that messages may arrive out of order. For information about message ordering behavior, see the SQS documentation.

Multiple SQS Queues

You may use a different queue for each HITType that you configure with notifications.

Mechanical Turk does not provide the ability to route events within a HITType to different queues. For example, you might prefer to have AssignmentSubmitted events for a HITType delivered to a different queue than HITReviewable events for that same HITType. Mechanical Turk will publish both events to the same queue. You can split the events into different queues by running an SQS client that pulls the messages and republishes them to different queues depending on the event type.

SQS Message Payload

The body of each SQS message is a JSON-encoded structure that provides support for multiple events in each message.

The JSON-encoded structure contains the following:

- EventDocVersion: This is the requested version that is passed in the call to SetHITTypeNotification (p. 129), such as '2006-05-05'. For a requested version, Mechanical Turk will not change the structure or definition of the output payload structure in a way that is not backward-compatible.
- EventDocId: A unique identifier for the Mechanical Turk. In rare cases, you may receive two different SQS messages for the same event, which can be detected by tracking the EventDocId values you have already seen.
- CustomerId: Your customerId.
- Events: A list of Event structures, described next.

The Event structure contains the following:

- EventType: A value corresponding to the EventType value in the notification specification data structure.

- EventTimestamp: A dateTime in the Coordinated Universal Time time zone, such as 2005-01-31T23:59:59Z.
- HITTypeId: The HIT type ID for the event.
- HITId: The HIT ID for the event.
- AssignmentId: The assignment ID for the event, if applicable.

Double Delivery

Amazon SQS already provides a 'MessageId' value that enables double-delivery detection in the typical SQS case. However, when receiving messages from Mechanical Turk, we recommend that you use the EventDocId value for double-delivery detection. This will cover an additional scenario in which you may see the same EventDocId in two messages with distinct MessageIds.

Most messages are safe to process twice, since they represent independent one-way state changes. Consider whether detection of repeated messages is important for your application. You may be able to simply process the message and ignore it if it appears to have been applied already.