Document 1

ATTACHMENT B

Outlook

s22 ---21/08/2025 01:37:43 PM---Approval of changes to the Coding Service user guide for the

s22 ---21/08/2025 01:37:43 PM---Approval of changes to the Coding Service user guide for the website [SEC=OFFICIAL]

Approval of changes to the Coding Service user guide for the website [SEC=OFFICIAL]

WoAG and ABS Coding Capability WDB s22 21/08/2025 01:37 PM

*

Basics

s22

s22 @abs.gov.au

21/08/2025 01:37 PM

Send To s22 @abs.gov.au>

cc s22 @abs.gov.au> s22 @abs.gov.au>, "WoAG and ABS Coding Capability WDB" <woag.and.abs.coding.capability.wdb@abs.gov.au>

Subject Approval of changes to the Coding Service user guide for the website [SEC=OFFICIAL]

Protective Mark

Information management markers	Caveat
Categories	Project Documentation\User guides and registration
Visibility	
Editors	
Readers	
Document Usage	Error - Disposal Authority not set - determines recordkeeping action
Document Id	DCOO-DKR6CT

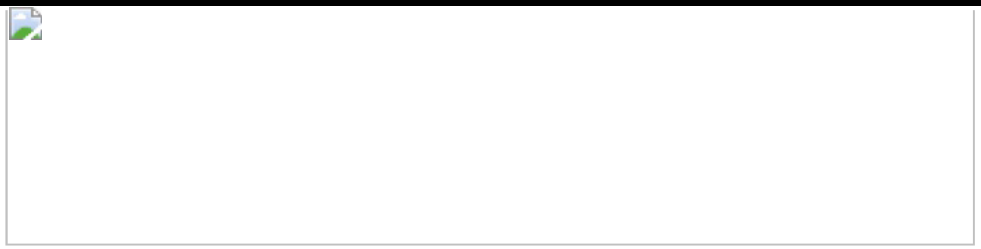
Hi s22 could you please approve the changes in the attached document by CoB Monday 25 August?

I have also attached an Integration Script Examples mini guide that s22 and s22 put together for me, to assist non-technical people to work with their technical teams to understand what is involved in integrating the Coding Service. This mini-guide is linked at the end of the User Guide as an additional resource.

Thank you

s22

s22 (she/her)
Stakeholder Relationship Manager, Census and WoAG Coding Capability Project
Business Register and Statistical Standards Branch| **Australian Bureau of Statistics**
(E) s22 @abs.gov.au **(W)** www.abs.gov.au



(See attached file: User Guide changes for August 2025 1408.docx)(See attached file: Integration script examples.docx)

Outlook

RE: Approval of changes to the Coding Service user guide for the website [SEC=OFFICIAL]

From s22 @abs.gov.au>
Date Fri 2025-08-29 14:09
To s22 @abs.gov.au>
Cc s22 @abs.gov.au>; s22 @abs.gov.au>; WoAG and ABS Coding Capability WDB <woag.and.abs.coding.capability.wdb@abs.gov.au>; s22 <s22 @abs.gov.au>

Oh perfect, thanks s22 😊

s22

s22

Stakeholder Relationship Manager, Census and WoAG Coding Capability Project

Business Register and Statistical Standards Branch| **Australian Bureau of Statistics**

(E) s22 **@abs.gov.au (W) www.abs.gov.au**

The ABS Privacy Policy outlines how the ABS handles any personal information that you provide to us.

The Australian Bureau of Statistics acknowledges the Traditional Custodians of Country throughout Australia and recognises their continuing connection to land, waters and community. We pay our respects to their Elders, past and present, their histories and cultures.

From: s22
Sent: Friday, 29 August 2025 12:16 PM
To: s22
Cc: s22 ; s22 ; WoAG and ABS Coding Capability WDB
Subject: Re: Approval of changes to the Coding Service user guide for the website [SEC=OFFICIAL]

Hi s22
Apologies for not sending this back sooner. I totally forgot.
The documents look good and see no obvious issues.
Consider this approved.
Cheers

s22

s22

TSD Census | Technology Services Division **| Australian Bureau of Statistics**

(P) s22 **(M)** s22

(E) s22 **@abs.gov.au (W) www.abs.gov.au**

From: s22 @abs.gov.au>
Sent: Thursday, August 21, 2025 11:37
To: s22 @abs.gov.au>
Cc: s22 @abs.gov.au>; s22

s22 @abs.gov.au>; WoAG and ABS Coding Capability WDB <woag.and.abs.coding.capability.wdb@abs.gov.au>
Subject: Approval of changes to the Coding Service user guide for the website [SEC=OFFICIAL]

Hi s22 could you please approve the changes in the attached document by CoB Monday 25 August?

I have also attached an Integration Script Examples mini guide that s22 and s22 put together for me, to assist non-technical people to work with their technical teams to understand what is involved in integrating the Coding Service. This mini-guide is linked at the end of the User Guide as an additional resource.

Thank you

s22

s22

Stakeholder Relationship Manager, Census and WoAG Coding Capability Project

Business Register and Statistical Standards Branch| **Australian Bureau of Statistics**

(E) s22 @abs.gov.au **(W)** www.abs.gov.au

The ABS Privacy Policy outlines how the ABS handles any personal information that you provide to us.

The Australian Bureau of Statistics acknowledges the Traditional Custodians of Country throughout Australia and recognises their continuing connection to land, waters and community. We pay our respects to their Elders, past and present, their histories and cultures.

Document 3

WoAG Occupation Coding Service User Guide – Changes for website

Details the API endpoints available for the Coding Service, and provides access and integration instructions.

Released

30/06/2025

Introduction

The Whole-of-Australian-Government (WoAG) Occupation Coding Service ('the Coding Service' or 'the service') ~~has been~~was designed by the Australian Bureau of Statistics (ABS) to provide a single occupation coder across government, industry and the community.

The Coding Service will code occupation data to the latest Australian standard occupation classification titles and codes.

Design

The Coding Service ~~has been~~was built with supervised machine-learning technology~~,~~ to train hierarchical support vector machine (HSVM) models, that provide high-quality and comprehensive automated coding against hierarchical classification categories. A confidence threshold is applied to the service, ensuring that outputs are high quality.

The service is called via an Application Programming Interface (API), designed to support integration across systems and platforms (including online forms and survey instruments) by offering authenticated, standards-compliant endpoints.

All API services are hosted in Australia to comply with relevant data sovereignty and privacy regulations.

Users have the option to register as a public user (throttled service), or a partner user (enhanced capability). Public and partner registrations enable the following services:

Public user

- Single record (synchronous) coding
- Small batch synchronous coding (up to 300 records)

Partner user

- Single record (synchronous) coding
- Small batch synchronous coding (up to 300 records)

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, English (Australia)

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, English (Australia)

Formatted: Font: (Default) Arial

- Large-file asynchronous upload/download bulk coding (from 1 record to millions of records)

Public user real time coding API calls will be throttled at 1 request per second, with ~~in~~ a ceiling of 1,000 requests per day. Partner user API calls will be throttled at 1 request per second, with a ceiling of 10,000 requests per day. Limits can be increased on a case-by-case basis (email coding capability@abs.gov.au in the first instance).

Security and technology standards

The Coding Service and API have been security assessed by an independent registered assessor within the Australian Signals Directorate (ASD) Information Security Registered Assessors Program (IRAP) Program. This assessment found the Coding Service and API to have met the control and security objectives defined through the Australian Government Information Security Manual (ISM).

The service has been built to comply to the ISM and the Protective Security Policy Framework (PSPF). It leverages modern web API technologies in accordance with the Australian Government's API Standard and globally recognised security frameworks. These standards ensure that the service is designed for safe, scalable integration across government and public-facing systems.

Both public and partner service users will be registered, and will be provided with relevant authorisation tokens to access the service.

See Coding Service security for more detail, including security controls to assist partner agencies in assessing their risks when using this service.

Using the guide

This user guide supports access to and use of the WoAG Occupation Coding Service. It is targeted toward software developers and technical professionals integrating the service into a client application.

The guide outlines the API endpoints available for accessing and using the service, and provides integration instructions for calling the API. It is structured to be followed sequentially from Getting started through to Gathering parameters.

Users will then proceed to Real-time (synchronous) coding for single record or small batch coding, or Asynchronous batch coding, depending on the data to be coded.

Synchronous coding

- The synchronous single-record coding service is designed for real-time usage (~1 second per record). It is suitable for small volumes and live systems such as

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

online forms and web surveys. Synchronous coding also supports coding small batches of records (up to 300 records) with similar per-record timing.

- Note: This service is not optimised for large volumes and should not be used for high-throughput workloads. Use asynchronous coding for scalable batch processing.

Asynchronous coding

- Asynchronous batch coding is designed for large datasets (from a single record to millions of records). While asynchronous coding is the most efficient service for larger batches of data, it is not real-time, and may be queued during high load periods.
- Batch uploads are submitted via the API, and status is checked via polling (operation endpoints).
- Response times for batch requests may range from a few minutes to several hours depending on file size, system demand and current queue load at the time of submission.

Getting started

WoAG Occupation Coding Service User Guide

Coding Service access instructions.

Released

30/06/2025

Pre-testing readiness

Before accessing the service, you will need to register for the coding service (see Registration). You will also need to consider the following:

- What you need to set up to pass the API packet to your API endpoint (url).
- What data you are going to code.
- Whether you need single record coding, small batch coding (up to 300 records), or large batch coding.
- Whether you need to ~~clean~~reformat ~~the~~ your data ~~first~~ (for example, batch data will need to fit specific formats, and you may want to add record identifiers to map back to your dataset).

See also Coding Service formats.

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Terms of Use

The use of the Coding Service API is governed by the Coding Service Terms of Use (which includes Service Level Expectations). All API users will be required to accept these Terms of Use prior to gaining access to the service.

Users requesting access to the service must be appropriately authorised to accept the Terms of Use on behalf of their organisation.

Registration

To register for the service and request client credentials, please read and accept the Terms of Use and complete the Registration Form.

Once you have registered, your ABS contact will email you:

- A client identifier, clientID.
- A client secret, clientSecret. This must be kept confidential as it is used when authenticating your requests.

The ABS will monitor registrations for usage, and users will be notified via email if their access is under review.

Authentication

WoAG Occupation Coding Service User Guide

Authentication information and instructions.

Released

30/06/2025

An authentication token is required to use the Coding Service. This is a unique, time-limited access key which is used to authenticate all API calls to the service.

Get an authentication token

To get an authentication token, you must first call the service's authentication endpoint with an authorisation header. More details are available in the AWS documentation, but the key input parameters are described below.

Note: Authentication is only required once every hour.

Do not include an authentication script in every API call. Too many token requests may result in an error. This error may also occur for you if there have been excessive

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

authentication requests from anyone else in the organisation. See Session Token Usage for more information.

Commented [s22]: Link to new section
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

Request syntax

POST /oauth2/token HTTP/1.1
Host: string
Content-Type: application/x-www-form-urlencoded
~~Authorisation~~Authorization: string

```
{
   grant_type: "client_credentials"
}
```

Request header parameters

Host
The host name for your chosen API. This will be either https://partner-coder.auth.abs.gov.au or https://public-coder.auth.abs.gov.au depending on whether you have registered for the partner or the public coding service.

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

Authorisation
Basic authorisation method with a base64 authorisation token (encodedAuthString), computed from the client ID and client secret provided upon registration.
encodedAuthString can be computed via the bash command:
$ echo -n "${clientID}:${clientSecret}" | base64
Type: String

Response syntax

HTTP/1.1 200 OK
Content-Type: application/json

```
{
   access_token: "string"
}
```

Response elements

access_token
Your unique access token which can be used to authenticate all API calls to the coding service.
Type: String

Examples

On registering for the coding service, this user was issued with the following:

- ClientID: “client1”
- ClientSecret: “secret123”

encodedAuthString should be the base64 encoding of “client1:secret123” and the entire request is as follows:

Sample Request

POST /oauth2/token HTTP/1.1
Host: https://partner-coder.auth.abs.gov.au
Content-Type: application/x-www-form-urlencoded
~~Authorisation~~Authorization: Basic Y2xpZW50MTpzZWNyZXQxMjM=

```
{
   grant_type: "client_credentials"
}
```

Sample Response

HTTP/1.1 200 OK
Content-Type: application/json

```
{
   access_token: "example token"
}
```

See Integration script examples for an example PowerShell script to request an authentication token.

Use an authentication token

To authenticate against the coding API service, you will need to include your access token in the header of any API calls.

- Your token will last one hour from the time of issue, after which you will need to request a new token.

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

- You do not need to request a new token for each API call.

The API calls are of a short duration, usually less than a few seconds. When initiated, each call will check the authentication and then continue with the rest of the call. If the call was approved at the start, it will return a response if the timer runs out.

Asynchronous batch calls may take longer to return results, and you may have to re-authorise to receive the results.

Request header parameters

Authorization

The authorisation token retrieved via the Authentication mechanism.
Type: String

Host

The host name for your chosen API. This will be either https://partner-coder.api.abs.gov.au or https://public-coder.api.abs.gov.au depending on whether you have registered for the partner or the public coding service.
Type: String

See Integration script examples for an example PowerShell token request script.

Session token usage

For ~~efficiency and~~ optimal performance, ~~integrators should~~ please reuse a single session token per hour and avoid hitting service limits by caching tokens. Single session tokens will be valid for all users interacting with webforms or interfaces during that time.

- Token Type: AWS Cognito M2M session token
- Token Validity: 60 minutes
- Usage Scope: Expect shared use of a single token across all users of a webform/integration/process
- Storage: Store locally in your backend (e.g., in-memory, file, cache)
- Rate Guidance: Max 1 token request per hour per webform

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, Strikethrough
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, Strikethrough
Formatted: Font: (Default) Arial

Exceeding this limit may result in the service provided to you being blocked or degraded. Restrictions are time limited, and requests for authorisation during this time will result in http 403 errors.

Please ensure that any scripts written for non-technical people to copy and paste for coding single records or running small batches do not include a token call. Provide authentication scripts separately to be used at need.

Recommended integration pattern

1. Backend checks for existing token
2. If token is valid, reuse it
3. If token is expired or missing, request a new one
4. Use the token to call the API for all users of the form

Token re-use flow diagram

Recommended flow for session token re-use across users of a webform.

Coding service formats

WoAG Occupation Coding Service User Guide

Request formats and recommended text inputs.

Released

30/06/2025

Request formats

The coding service uses JSON format for the following services:

- Real-time (synchronous) public coding service
- Real-time partner coding service
- Real-time small batch coding service

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

It uses JSONL format for the large batch/bulk (asynchronous) partner coding service.

The GET Data method will return the following for each of the specified services for occupation coding:

Download

Service	Returns
Real-time (synchronous) public coding service	• One or more classification codes and titles for the free text supplied
Real-time partner coding service	• One or more classification codes and titles for the free text supplied
Real-time small batch coding service (up to 300 records)	• The best match 1-digit to 6-digit codes and titles (moving up the classification hierarchy from 6-digit to 1-digit level) for the free text supplied • If the coder cannot code the free text supplied, it will provide 3 suggestions
Large batch/bulk (asynchronous) partner coding service	• The best match 1-digit to 6-digit codes and titles (moving up the classification hierarchy from 6-digit to 1-digit level) for the free text supplied • If the coder cannot code the free text supplied, it will provide 3 suggestions

Recommended text input for coding

- The occupation coder will perform optimally when provided with both a job title and tasks as free text inputs, as this is how the ML training was carried out.
- Large batch (asynchronous) coding requires both the Occp_text and Tasks_text fields in the call. If input text for one or other of the fields is not available, include the blank field. For example:
  "occp_text": "sewing machinist"
  "tasks_text": ""
- The coder will not perform as well with just one text field ~~entered~~ completed (i.e if only the job title or only the task text is entered). If results are unsuccessful, entering more information will help the ~~Coding S~~service make better predictions.
- For synchronous coding, t~~T~~ext strings can be a maximum of 100 characters only (a total of 100 characters for combined occupation and task input text entries).

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: 12 pt
Formatted: Font: (Default) Arial

- For asynchronous coding, text strings can be a total of 300 characters for the combined fields.
- The coding service API will not accept custom data queries or query string parameters.
- The service does not recognise classification codes as inputs, therefore it is not possible to recode a dataset from, for example, a six-digit ANZSCO code to a six-digit OSCA code. Datasets with just ANZSCO codes may be able to be recoded to OSCA if the ANZSCO occupation title is reattached to each record. Including more information in the tasks text, such as the occupation descriptions, will give better outcomes.
- The coding service has been trained on English inputs only. The service accepts printable ASCII characters, which includes all English letters and connectives, but excludes certain accents, foreign currency symbols and control characters like file endings or backspace. Including a bad character may result in an 'Invalid request body' error.

Contextual considerations

The contextual assumption of the input text is that the text relates to and describes a person's job. The coder is able to recognise a very broad vocabulary and will attempt to code all input text, regardless of context, so users need to ensure a contextual fit between their input data and the coding task being undertaken.

For example, if a person describes their job as a 'prisoner', the Coding Service assumes a context that the ~~occupation~~ job to be coded works with prisoners in some way, and codes to 'Correctional Officer'. Likewise, the input text 'baby' codes to 'Nanny'.

Multiple occupation entries

The service is designed to provide a single occupation code and title for a single ~~occupation~~ record. If multiple ~~occupations~~ jobs per record are entered in the ~~occupation~~ job title text input, the coder will attempt to code the provided text to a best fit single occupation code at the most detailed level.

The output will reflect the training data and will depend on how many times the two jobs were present together in the training data. The Coding Service will default to whatever is most commonly found in the training data.

- If multiple ~~occupations~~ jobs are present, you will need to format each job as a separate request.

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, Bold

Formatted: Font: (Default) Arial

API Endpoints and HTTP methods

WoAG Occupation Coding Service User Guide

Coding Service endpoints and methods.

Released

30/06/2025

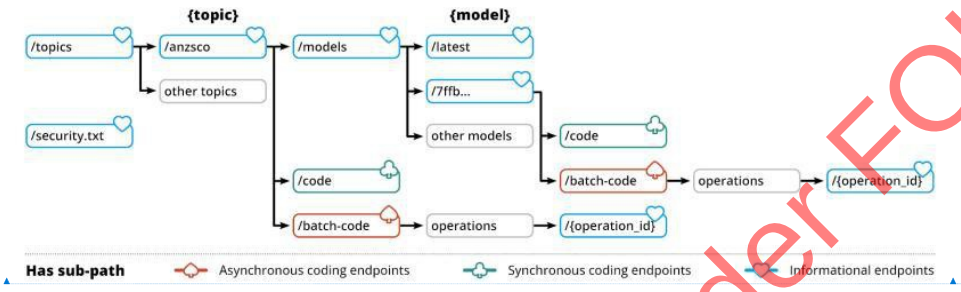
The API endpoints and their HTTP methods are outlined below in both the table and the diagram.

Download

Endpoint	Description	HTTP verb
/v1/topics	Retrieve a list of available topics.	GET
/v1/topics/{topic}	Describes the input format for the given topic.	GET
/v1/topics/{topic}/code	Synchronously codes a single record or small batch of records against the latest model for a given topic.	POST
/v1/topics/{topic}/models	Lists the available models and their input formats for a given topic.	GET
/v1/topics/{topic}/models/latest	Describes the input format for the latest model for a given topic.	GET
/v1/topics/{topic}/models/{model}/code	Synchronously codes a single record or small batch of records against a specific model for the given topic.	POST
/v1/topics/{topic}/batch-code	Creates a new asynchronous batch inference operation against the latest model for a given topic.	POST
/v1/topics/{topic}/models/{model}/batch-code	Creates a new asynchronous batch inference operation against a specific model for the given topic.	POST
/v1/topics/{topic}/batch-code/operations/{operation_id}	Checks the status of a batch inference operation.	GET
/v1/topics/{topic}/models/{model}/batch-	Checks the status of a batch inference operation.	GET

Formatted: Font: (Default) Arial

Endpoint	Description	HTTP verb
code/operations/{operation_id}		
/v1/security.txt	Returns contact details for reporting issues.	GET



Gathering parameters

WoAG Occupation Coding Service User Guide

Information on topics and models.

Released

30/06/2025

Listing available topics

Before coding against a topic (classification), you must confirm that the topic is supported by the application. You will also need to record its corresponding uriName for further calls to the API.

Request syntax

GET /v1/topics HTTP/1.1
Host: string
Content-type: application/json
~~Authorisation~~Authorization: string

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

URI request parameters

The request does not use any URI parameters.

Request body

The request does not have a request body.

Response syntax

```
HTTP/1.1 200 OK
Content-type: application/json
[
   {
      "uriName": "string",
      "fullName": "string"
   }
]
```

Response elements

If the action is successful, the service sends back an HTTP 200 response. The API returns an array of Topic objects representing all the coding topics currently supported by the API.

Errors

For information about the errors that are common to all actions, see Errors and suggested actions.

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Examples

Sample request

```
GET /v1/topics HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-type: application/json
AuthorisationAuthorization: example token
```

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Sample response

```
HTTP/1.1 200 OK
Content-type: application/json
[
   {
      "uriName": "osca",
     "fullName": "OSCA - Occupation Standard Classification for Australia"
```

```
    }
]
```

Getting the input format for the latest model for a topic

Different models are coded against different input formats. If you are using the latest (default) model for your specified topic, you can get the input format via the following mechanism.

If you are using another model, the input format will be provided as part of the list of available models.

Request syntax

```
GET /v1/topics/{topic}/models/latest HTTP/1.1
Host: string
Content-type: application/json
~~Authorisation~~Authorization: string
```

URI request parameters

Download

URI request parameters

The uriName for the coding topic of interest. This can be acquired by listing **topic** the available topics.
Required: Yes

Request body

The request does not have a request body.

Response syntax

```
HTTP/1.1 200 OK
Content-type: application/json
{
   "modelId": "string",
   "modelVersion": number,
   "modelReleaseDate": "string",
   "modelType": "string",
   "inputFormat":  [
      "string"
   ],
   "topicStandard": "string",
```

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

```
    "topicVersion": "string"
}
```

Response elements

If the action is successful, the service sends back an HTTP 200 response. The API returns a Model object representing the latest model for the given topic, which includes the expected input format.

Errors

For information about the errors that are common to all actions, see Errors and suggested actions.

Example

Getting the latest occupation model:

Sample request

```
GET /v1/topics/osca/models/latest HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-type: application/json
~~Authorisation~~Authorization: example token
```

Sample response

```
HTTP/1.1 200 OK
Content-type: application/json
{
  "modelId": "00000000-0000-0000-0000-000000000000",
  "modelVersion": 2,
  "modelReleaseDate": "2023-12-20T06:06:44.514Z",
  "modelType": "hsvm2",
  "inputFormat": [
     "occp_text",
     "tasks_text"
   ],
  "topicStandard": "OSCA - Occupation Standard Classification for Australia",
  "topicVersion": "2024"
}
```

Listing available models for a given topic

To code against a specific machine learning model, you can browse the available models and their input formats by calling this endpoint.

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Request syntax

GET /v1/topics/{topic}/models HTTP/1.1
Host: string
Content-type: application/json
~~Authorisation~~Authorization: string

URI request parameters

Download

URI request parameters

The uriName for the coding topic of interest. This can be acquired by listing **topic** the available topics.
Required: Yes

Request body

The request does not have a request body.

Response syntax

```
HTTP/1.1 200 OK
Content-type: application/json
[
   {
      "modelId": "string",
      "modelVersion": number,
      "modelReleaseDate": "string",
      "modelType": "string",
      "inputFormat": [
         "string"
      ],
      "topicStandard": "string",
      "topicVersion": "string"
   }
]
```

Response elements

If the action is successful, the service sends back an HTTP 200 response. The API returns either a SynchronousCodeResponse object or an array of SynchronousCodeResponse objects corresponding to the input records.

Errors

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

For information about the errors that are common to all actions, see Errors and suggested actions. The following errors may occur when calling this service:

No models found for topic {topic} : No models are available for the provided topic parameter. Please reach out to the ABS to investigate why no model is available. HTTP Status Code: 500 (Internal Server Error)

Example

Listing all ANZSCO models:

Sample request

GET /v1/topics/anzsco/models HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-type: application/json
~~Authorisation~~Authorization: example token

Sample response

HTTP/1.1 200 OK
Content-type: application/json


```
[
  {
    "modelId": "00000000-0000-0000-0000-000000000002",
    "modelVersion": 2,
    "modelReleaseDate": "2023-12-20T06:06:44.514Z",
    "modelType": "hsvm2",
     "inputFormat": [
       "occp_text",
       "tasks_text"
     ],
   "topicStandard": " ANZSCO - Australian and New Zealand Standard Classification of Occupations",
    "topicVersion": "2022"
  },
  {
    "modelId": "00000000-0000-0000-0000-000000000001",
    "modelVersion": 1,
    "modelReleaseDate": "2025-05-20T06:06:44.514Z",
    "modelType": "hsvm2",
    "inputFormat": [
       "occp_text",
       "tasks_text"
```

```
    ],
    "topicStandard": " ANZSCO - Australian and New Zealand Standard Classification of Occupations",
    "topicVersion": "2022"
  }
]
```

Real-time (synchronous) coding

WoAG Occupation Coding Service User Guide

Single record and small batch coding.

Released

30/06/2025

Single record coding

The coding service has been designed to apply a classification code and title to a free text entry. The single record coding feature will enable public facing webforms and other points of data collection to have codes and titles suggested in real time (~1 second).

The service will provide multiple responses within a classification category as long as at least one response has a confidence level above the threshold.

It may only provide one response (for example, if there is only one six-digit code in that category).

If no responses are above the confidence threshold, the service will not return any results.

Small batch coding

A small JSON file of up to 300 text records can also be coded synchronously.

Note: when you are running a synchronous small batch, the whole packet needs to be syntactically correct. If the syntax fails, the whole batch will fail. As the operation is combined for the whole group of records, none of the records will be able to be coded if there is an error in any record.

When to use synchronous or asynchronous coding

Synchronous coding should only be used for single record coding or small batches of data. If you are coding 900 records, for example, it will be possible to run them in three small batch submissions.

Formatted: Font: (Default) Arial, Bold

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, Bold

Formatted: Font: (Default) Arial

Asynchronous large batch coding is recommended if you need to code or recode a large volume of data. (Large batch coding can be used to code from 1 record to millions of records.)

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, Italic

Formatted: Font: (Default) Arial

Coding against the latest model for a topic

This endpoint is used to code a single or small batch of free text records against the specified coding topic, using the latest model for that topic.

Request syntax

Depending on whether you are coding a single record or a small batch of records, your request will follow one of the following formats:

1. Coding a single free text record

```
POST /v1/topics/{topic}/code HTTP/1.1
Host: string
Content-type: application/json
~~Authorisation~~Authorization: string
{
   "record": {
      "occp_text": "string",
      "tasks_text": "string"
   },
   "numberOfSuggestions": number
}
```

2. Coding a small batch of free text records

```
POST /v1/topics/{topic}/code HTTP/1.1
Host: string
Content-type: application/json
~~Authorisation~~Authorization: string
{
   "records": [
      {
         "recordId": "string",
         "occp_text": "string",
         "tasks_text": "string"
      },
   ],
```

"numberOfSuggestions": number
}

URI request parameters

Download

URI request parameters

topic	The uriName of the topic against which the record is coded. This can be acquired by listing the available topics. Required: Yes

Request body

Download

Request body

record	The free text record to be coded. Type: Record object, following the input format specified by the model. Required: No, but either record or records must be provided.
records	The free text records to be coded. Type: Array of Record objects, following the input format specified by the model. Each item may optionally specify an additional string value recordId. Length Constraints: Minimum length of 1. Maximum length of 300. Required: No, but either record or records must be provided.
numberOfSuggestions	The number of suggested codes to be provided if the record cannot be coded successfully. The maximum value of this field is 16. Type: Number Required: No

Response syntax

The response of this endpoint will depend on whether your input request contained a single record or a small batch of records.

1. Coding a single free text record

```
HTTP/1.1 200 OK
Content-type: application/json
{
   "codeStatus": "string",
   "input": {
      "occp_text": "string",
      "tasks_text": "string"
   },
   "result": [
      {
         "codeCategory": "string",
         "codeLabel": "string",
         "codeConfidence": number
      }
   ],
}
```

2. Coding a small batch of free text records

```
HTTP/1.1 200 OK
Content-type: application/json
[
   {
      "recordId": "string",
      "codeStatus": "string",
      "input": {
         "occp_text": "string",
         "tasks_text": "string"
      },
      "result": [
         {
            "codeCategory": "string",
            "codeLabel": "string",
            "codeConfidence": number
         }
      ],
   }
]
```

Response elements

If the action is successful, the service sends back an HTTP 200 response. The API returns either a SynchronousCodeResponse object or an array of SynchronousCodeResponse objects corresponding to the input records.

Errors

For information about the errors that are common to all actions, see Errors and suggested actions. The following errors may occur when calling this service:

Examples

Successfully coded a single record using only one free text field:

Sample request

```
POST /v1/topics/osca/code HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-type: application/json
~~Authorisation~~Authorization: example token
{
   "record": {
      "occp_text": "Software developer. Writes code, tests"
   },
   "numberOfSuggestions": 3
}
```

Sample response

```
HTTP/1.1 200 OK
Content-type: application/json
{
   "codeStatus": "successful",
   "input": {
      "occp_text": "Software developer. Writes code, tests"
   },
   "result": [{
      "codeCategory": "261313",
      "codeLabel": "Software Engineer",
      "codeConfidence": 0.6093962788581848
   },{
```

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Space After: 0 pt

```
        "codeCategory": "261312",
        "codeLabel": "Developer Programmer",
        "codeConfidence": 0.43940293565392494
    },{
        "codeCategory": "261399",
        "codeLabel": "Software and Applications Programmers nec",
        "codeConfidence": 0.43756575286388397
    }]
}
```


Successfully coded a single record using all free text fields:

Sample request

```
POST /v1/topics/osca/code HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-type: application/json
AuthorisationAuthorization: example token
{
    "record": {
        "occp_text": "software developer",
       "tasks_text": "writing code and unit tests"
    },
    "numberOfSuggestions": 3
}
```

Sample response


```
HTTP/1.1 200 OK
Content-type: application/json
{
    "codeStatus": "successful",
    "input": {
        "occp_text": "software developer",
       "tasks_text": "writing code and unit tests"
    },
    "result": [{
        "codeCategory": "261313",
        "codeLabel": "Software Engineer",
        "codeConfidence": 0.6093962788581848
    },{
        "codeCategory": "261312",
```


Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

```
      "codeLabel": "Developer Programmer",
      "codeConfidence": 0.43940293565392494
   },{
      "codeCategory": "261399",
      "codeLabel": "Software and Applications Programmers nec",
      "codeConfidence": 0.43756575286388397
   }]
}
```

Unsuccessfully coded a single record using only one free text field:

Sample request

```
POST /v1/topics/osca/code HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-type: application/json
~~Authorisation~~Authorization: example token
{
   "record": {
      "occp_text": "Software developer. Writes code, tests"
   },
   "numberOfSuggestions": 3
}
```

Sample response

```
HTTP/1.1 200 OK
Content-type: application/json
{
   "codeStatus": "unsuccessful",
   "input": {
      "occp_text": "Software developer. Writes code, tests"
   },
   "result": [ ]
}
```

Coding a small batch of records:

Sample request

```
POST /v1/topics/osca/code HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
```

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

```
Content-type: application/json
~~Authorisation~~Authorization: example token
{
   "records": [
      {
         "occp_text": "software developer",
         "tasks_text": "writing code and unit tests"
      }, {
         "occp_text": "Paramedic, respond to emergencies"
      }, {
         "recordId": "1",
         "occp_text": "Sales assistant"
      }
   ],
   "numberOfSuggestions": 3
}
```

Sample response

```
HTTP/1.1 200 OK
Content-type: application/json
[
   {
      "codeStatus": "successful",
      "input": {
        "occp_text": "software developer",
         "tasks_text": "writing code and unit tests"
      },
      "result": [{
         "codeCategory": "261313",
         "codeLabel": "Software Engineer",
         "codeConfidence": 0.6093962788581848
      },{
         "codeCategory": "261312",
         "codeLabel": "Developer Programmer",
         "codeConfidence": 0.43940293565392494
      },{
         "codeCategory": "261399",
         "codeLabel": "Software and Applications Programmers",
         "codeConfidence": 0.43756575286388397
      }]
```

```
    }, {
        "codeStatus": "unsuccessful",
        "input": {
            "occp_text": "Paramedic, respond to emergencies"
        },
        "result": [ ]
    }, {
        "recordId": "1",
        "codeStatus": "unsuccessful",
        "input": {
            "occp_text": "Sales assistant"
        },
        "result": [ ]
    }
]
```

Coding against a specific model

This endpoint is used to code a single or small batch of free text records against the specified coding topic, using the specified model.

Request syntax

Depending on whether you are coding a single record or a small batch of records, your request will follow one of the following formats:

1. Coding a single free text record against a specific model

```
POST /v1/topics/{topic}/models/{model}/code HTTP/1.1
Host: string
Content-type: application/json
AuthorisationAuthorization: string
{
    "record": {
        "occp_text": "string",
        "tasks_text": "string"
    }
    "numberOfSuggestions": number
}
```

2. Coding records against a specific model

```
POST /v1/topics/{topic}/models/{model}/code HTTP/1.1
Host: string
Content-type: application/json
AuthorisationAuthorization: string
{
   "records": [
      {
         "recordId": "string",
         "occp_text": "string",
         "tasks_text": "string"
      }
   ],
   "numberOfSuggestions": number
}
```

URI request parameters

Download

URI request parameters

topic	The uriName of the topic against which the record is coded. This can be acquired by listing the available topics. Required: Yes
model	The model GUID for the model you would like to use to code records. This can be acquired by listing the available models for your topic. Required: Yes

Request body

Download

Request body

record	The free text record to be coded. Type: Record object, following the input format specified by the mode Required: No, but either record or records must be provided.
records	The free text records to be coded. Type: Array of Record objects, following the input format specified by may optionally specify an additional string value recordId.

Formatted Table
Formatted: Font: (Default) Arial

	Length Constraints: Minimum length of 1. Maximum length of 300. Required: No, but either record or records must be provided.
numberOfSuggestions	The number of suggested codes to be provided if the record cannot be coded successfully. The maximum value of this field is 16. Type: Number Required: No

Response syntax

The response of this endpoint will depend on whether your input request contained a single record or a small batch of records.

1. Coding a single free text record

```
HTTP/1.1 200 OK
Content-type: application/json
{
   "codeStatus": "string",
   "input": {
      "occp_text": "string",
      "tasks_text": "string"
   },
     "result": [
      {
         "codeCategory": "string",
         "codeLabel": "string",
         "codeConfidence": number
      }
   ],
}
```

2. Coding a small batch of free text records

```
HTTP/1.1 200 OK
Content-type: application/json
[
   {
      "recordId": "string",
      "codeStatus": "string",
      "input": {
         "occp_text": "string",
         "tasks_text": "string"
```

```
    },
    "result": [
      {
        "codeCategory": "string",
        "codeLabel": "string",
        "codeConfidence": number
      }
    ],
  }
]
```

Response elements

If the action is successful, the service sends back an HTTP 200 response. The API returns either a SynchronousCodeResponse object or an array of SynchronousCodeResponse objects corresponding to the input records.

Errors

Download

For information about the errors that are common to all actions, see Errors and suggested actions. The following errors may occur when calling this service:

Errors

Malformed record found in request	The free text input did not match the expected format for the model. You can check what the expected format is for a given topic here. HTTP Status Code: 400 (Bad Request)
Batch input contained no records	You tried to code a small batch of records but the records array was empty. Check that you have provided at least one record to be coded and that your request body is correctly formatted. HTTP Status Code: 400 (Bad Request)
Batch records exceeds length limit of 300	You tried to code too many records at once using the synchronous small batch service. Retry with a smaller batch size, or consider using the asynchronous batch coding service. HTTP Status Code: 400 (Bad Request)
There are record(s) outside	One or more records provided for synchronous coding had too many or too few characters. See Recommended text input for coding. The error will direct you to the problematic record(s) which should either be excluded or amended to meet the

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

the min or max char limit	character limits. HTTP Status Code: 400 (Bad Request)

Examples

Successfully coded a single record using all free text fields:

Sample request

```
POST /v1/topics/osca/models/GUID/code HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-type: application/json
~~Authorisation~~Authorization: example token
{
   "record": {
      "occp_text": "software developer",
      "tasks_text": "writing code and unit tests"
   },
   "numberOfSuggestions": 3
}
```

Sample response

```
HTTP/1.1 200 OK
Content-type: application/json
{
   "codeStatus": "successful",
   "input": {
      "occp_text": "software developer",
      "tasks_text": "writing code and unit tests"
   },
   "result": [{
      "codeCategory": "261313",
      "codeLabel": "Software Engineer",
      "codeConfidence": 0.6093962788581848
   },{
      "codeCategory": "261312",
      "codeLabel": "Developer Programmer",
      "codeConfidence": 0.43940293565392494
   },{
      "codeCategory": "261399",
      "codeLabel": "Software and Applications Programmers nec",
```

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

```
      "codeConfidence": 0.43756575286388397
   }]
}
```

Coding a small batch of records:

Sample request

```
POST /v1/topics/osca/models/GUID/code HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-type: application/json
~~Authorisation~~Authorization: example token
{
   "records": [
      {
         "occp_text": "software developer",
         "tasks_text": "writing code and unit tests"
      }, {
         "occp_text": "Paramedic, respond to emergencies"
      }, {
         "recordId": "1",
         "occp_text": "Sales assistant"
      }
   ],
   "numberOfSuggestions": 3
}
```

Sample response

```
HTTP/1.1 200 OK
Content-type: application/json
[
   {
      "codeStatus": "successful",
      "input": {
         "occp_text": "software developer",
         "tasks_text": "writing code and unit tests"
      },
      "result": [{
        "codeCategory": "261313",
         "codeLabel": "Software Engineer",
         "codeConfidence": 0.6093962788581848
```

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

```
        },{
            "codeCategory": "261312",
            "codeLabel": "Developer Programmer",
           "codeConfidence": 0.43940293565392494
        },{
            "codeCategory": "261399",
            "codeLabel": "Software and Applications Programmers",
            "codeConfidence": 0.43756575286388397
        }]
    }, {
        "codeStatus": "unsuccessful",
        "input": {
            "occp_text": "Paramedic, respond to emergencies"
        },
        "result": [ ]
    }, {
        "recordId": "1",
        "codeStatus": "unsuccessful",
        "input": {
            "occp_text": "Sales assistant"
        },
        "result": [ ]
    }
]
```


See Integration script examples for example PowerShell single record and small batch coding scripts.

Asynchronous batch coding

WoAG Occupation Coding Service User Guide

Coding a large volume of data.

Released

30/06/2025

In addition to real-time coding of single records and small batches of data, the Coding Service has been designed to code large datasets through asynchronous batching (that is, returning data after a short period of time).

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

The asynchronous service can be used for as little as one record, up to millions of records.

Note: Asynchronous batch coding should be used if you need to code or recode a large volume of data. While it is the most efficient method of coding larger datasets, it is not real-time, and may be subject to queueing during high load periods.

Getting an upload URL for input data to a batch coding operation

This endpoint is used to create an asynchronous batch inference operation. The API will return a location where you can upload your input file and begin your batch inference operation.

Request syntax

Depending on whether you are specifying a model against which to code your records, your request will follow one of the following formats:

1. Coding records against the latest model

POST /v1/topics/{topic}/batch-code HTTP/1.1
Host: string
Content-type: application/json
~~Authorisation~~Authorization: string

2. Coding records against a specific model

POST /v1/topics/{topic}/models/{model}/batch-code HTTP/1.1
Host: string
Content-type: application/json
~~Authorisation~~Authorization: string

URI request parameters

Download

URI request parameters

topic	The uriName of the topic against which the record is coded. This can be acquired by listing the available topics. Required: Yes
model	The model GUID for the model you would like to use to code records. This can be acquired by listing the available models for your topic. Required: No

Request body

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

The request does not have a request body.

Response syntax

```
HTTP/1.1 200 OK
Content-type: application/json
{
   "requestUploadUrl": "string",
   "operationId": "string",
   "bucketKmsKeyArn": "string"
}
```

Response elements

Download

If the action is successful, the service sends back an HTTP 200 response. The following data is returned in JSON format by the service:

Response elements

requestUploadUrl	A URL where the records file is to be uploaded. Type: String
operationId	The identifier of the operation, to be used to check the status of this job. This must be recorded at this point to maintain access to the operation. Type: String, in GUID format.
bucketKmsKeyArn	A parameter used by the ABS system to ensure the operation's input data is from the same user who created the operation. This must be passed into the x-amz-server-side-encryption-aws-kms-key-id header when uploading your input file. Type: String

Errors

For information about the errors that are common to all actions, see Errors and suggested actions.

Examples

Creating a new operation to code against the latest model for occupation:

Sample Request

Formatted Table

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

POST /v1/topics/osca/batch-code HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-Type: application/json
~~Authorisation~~Authorization: example token

Sample Response

HTTP/1.1 200 OK
Content-type: application/json
{
"requestUploadUrl": "https://domain/endpoint?queries",
"operationId": "00000000-0000-0000-0000-000000000000",
"bucketKmsKeyArn": "xyz"
}

Creating a new operation to code against a specified model:

Sample Request

POST /v1/topics/anzsco/models/GUID/batch-code HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-Type: application/json
~~Authorisation~~Authorization: example token

Sample Response

HTTP/1.1 200 OK

Content-type: application/json
{
"requestUploadUrl": "https://domain/endpoint?queries",
"operationId": "00000000-0000-0000-0000-000000000000",
"bucketKmsKeyArn": "xyz"
}

Uploading data for inference

Once you have created an inference operation, you will need to upload your data to the provided requestUploadUrl. This is a pre-signed HTTP request which is managed by the AWS S3 server, and the expected input is outlined below. Both Occp_text and Tasks_test fields are required (although one may be empty, indicated by "").

Request Syntax

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

PUT requestUploadUrl HTTP/1.1
x-amz-server-side-encryption: aws:kms
x-amz-server-side-encryption-aws-kms-key-id: string

{ "recordId": "string", "occp_text": "string", "tasks_text": "string" }
...

URI Request Parameters

Download

URI Request Parameters

requestUploadUrl	The location where the input file is being uploaded. This is provided when you first create the inference operation. Type: String

Request Header Parameters

Download

Please note: the x-amz-server-side-encryption header is not variable and should always have the value aws:kms.

Request Header Parameters

x-amz-server-side-encryption-aws-kms-key-id	A parameter used by the ABS system to ensure the input data is from the same user who created the operation. This is provided in the bucketKmsKeyArn field when you first create your inference operation. Type: String

Request Body

Download

The request accepts your input file in JSONL format. The maximum input file size is 5GB. All lines of input must contain the same fields, and these fields should satisfy the Record type for the relevant topic/model as specified when creating the upload URL. You may specify the additional field outlined below:

Request Body

recordId	An identifier for the record being coded. This need not be unique. Type: String Required: No

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

Response Syntax

HTTP/1.1 200 OK

Errors

For information about the errors that are common to all actions, see Errors and suggested actions.

Examples

Specifying ~~all free text inputs and~~ a record identifier:

Sample Request

PUT https://domain/endpoint?queries HTTP/1.1
x-amz-server-side-encryption: aws:kms
x-amz-server-side-encryption-aws-kms-key-id: xyz

{ "recordId": "1", "occp_text": "software developer", "tasks_text": "writing code and unit tests" }
{ "recordId": "2", "occp_text": "Paramedic", "tasks_text": "responding to medical emergencies" }

{ "recordId": "3", "occp_text": "Brickie", "tasks_text": "" }
...

Sample Response

HTTP/1.1 200 OK

~~**Specifying a single free text input and a record identifier:**~~

~~**Sample Request**~~

~~PUT https://domain/endpoint?queries HTTP/1.1~~
~~x-amz-server-side-encryption: aws:kms~~
~~x-amz-server-side-encryption-aws-kms-key-id: xyz~~

~~{ "recordId": "1", "occp_text": "software developer, writes code and unit tests" }~~
~~{ "recordId": "2", "occp_text": "Paramedic, respond to emergencies" }~~
~~...~~

~~**Sample Response**~~

~~HTTP/1.1 200 OK~~

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Specifying ~~all free text inputs and~~ no record identifier:

Sample Request

PUT https://domain/endpoint?queries HTTP/1.1
x-amz-server-side-encryption: aws:kms
x-amz-server-side-encryption-aws-kms-key-id: xyz

{ "occp_text": "software developer", "tasks_text": "writing code and unit tests" }
{ "occp_text": "Paramedic", "tasks_text": "responding to medical emergencies" }

{ "occp_text": "Brickie", "tasks_text": "" }

...

Sample Response

HTTP/1.1 200 OK

Checking the status of a batch inference operation

This endpoint is used to check the status of your batch inference job. When the status of your job is complete, the service will return a URL to copy into your web browser to retrieve your coded data.

Request Syntax

Depending on whether you are specifying a model against which to code your records, your request will follow one of the following formats. The application backend handles these requests identically, so you don't need to worry about recording the model which you used when you began the operation.

1. Checking an operation by specifying the topic only

GET /v1/topics/{topic}/batch-code/operations/{operation_id} HTTP/1.1
Host: string
Content-type: application/json
~~Authorisation~~Authorization: string

2. Checking an operation by specifying both the topic and model

GET /v1/topics/{topic}/models/{model}/batch-code/operations/{operation_id} HTTP/1.1
Host: string
Content-type: application/json
~~Authorisation~~Authorization: string

URI Request Parameters

Download

URI Request Parameters

topic	The uriName of the topic against which the record is coded. This can be acquired by listing the available topics. Required: Yes
model	The model GUID for the model you would like to use to code records. This can be acquired by listing the available models for your topic. Required: No
operation_id	The GUID of the operation to get the status of. This value is provided when you first create your inference operation. Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200 OK
Content-type: application/json
{
   "operationStatus": "string",
   "responseDownloadUrl": "string",
   "error": "string"
}
```

Response Elements

Download

If the specified operation exists, the service sends back an HTTP 200 OK status code. The status of the operation will dictate the contents of the response. This data is returned in JSON format by the service:

Response Elements

operationStatus	The status of the operation. Type: String Valid Values: awaiting_input | in_progress | complete | timed_out | failed

responseDownloadUrl	A URL where the output data file can be downloaded. This field is optional and is returned only if operationStatus is complete. Type: String
metadataDownloadUrl	A URL where the output metadata file can be downloaded. This file includes information about the model used to code your data.This field is optional and is returned only if operationStatus is complete. Type: String
error	Information on why the operation failed. This field is optional and is returned only if operationStatus is failed.

A note about presigned URLs

The responseDownloadUrl and metadataDownloadUrl are presigned URLs. Anyone with this link will be able to download your output file, so it is your responsibility to keep the link secret.

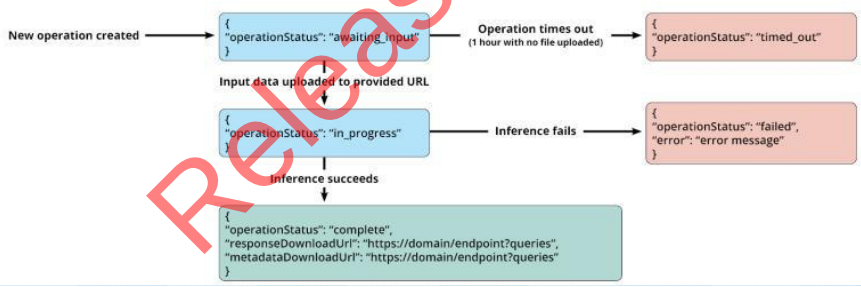
The link will expire after one hour, after which you will have to get a new URL for your output file.

Your output files will be deleted from the system within 24 hours after your inference operation succeeds.

A state machine indicating the progression of operations is shown below:

ImageDescription

View full screen



Errors

Download

For information about the errors that are common to all actions, see Errors and suggested actions. The following errors may occur when calling this service:

Errors

Unable to retrieve operation for given id	No operations were found to match the given operation_id. Please confirm your operation ID. If you have lost your operation ID, you will have to create a new operation. HTTP Status Code: 404 (Not Found)
User is not authorised to retrieve operation GUID	The specified operation does not belong to the current user. You may have authenticated with the wrong user or specified the wrong operation_id. Try authenticating again with the right credentials, and confirm your operation. HTTP Status Code: 401 (Unauthorised)

Examples

Getting the status of an operation:

Sample Request

GET /v1/topics/osca/batch-code/operations/GUID HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-type: application/json
~~Authorisation~~Authorization: example token

Sample Request specifying the model used

GET /v1/topics/anzsco/models/GUID/batch-code/operations/GUID HTTP/1.1
Host: https://partner-coder.api.abs.gov.au
Content-type: application/json
~~Authorisation~~Authorization: example token

Sample Responses

HTTP/1.1 200 OK
Content-type: application/json

and any of the following:

Download

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

Request sample	Expected response body	Interpretation
New operation (data not yet uploaded)	{ "operationStatus": "awaiting_input" }	• The server acknowledges the operation exists. • No input data file has yet been received. • The operation is pending your next action - typically an upload via PUT request.
Just uploaded	{ "operationStatus": "in_progress" }	• The server has received the input data file. • The specified operation is now running, or may be queued to run soon. • You should keep checking in periodically (for example, up to once every ten minutes) to see how the operation is progressing. • The output files will be deleted within 24 hours.
Never uploaded	{ "operationStatus": "timed_out" }	• The server acknowledges the operation exists. • No input data has yet been received. • The operation has timed out due to inactivity and can no longer accept input data. • If you wish to run an asynchronous batch operation, you will need to create a new operation.

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

Request sample	Expected response body	Interpretation
Operation complete	{ "operationStatus": "complete", "responseDownloadUrl": "https://domain/endpoint?queries", "metadataDownloadUrl": "https://domain/endpoint?queries", }	• The specified operation is now complete. • The output files are now available at the provided URLs. • You should download the output files now as they will be deleted within 24 hours. • There may be unsuccessfully coded records in the output file. Errors will be reported on a record-by-record basis where possible. This reduces the need to recode the entire input file.
Operation failed	{ "operationStatus": "failed", "error": "error message" }	• The specified operation has failed inference. • Check your input file for any errors or invalid records and try again. • The error message may provide context on what caused the operation failure. If the error message does not help resolve the issue, please note your operation id when contacting us for support. • If you wish to run another asynchronous batch operation, you will need to create a new operation.

Formatted Table

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Downloading processed data from a complete operation

Once your asynchronous inference operation is complete, you can download the output file by accessing (copying into a web browser) the responseDownloadUrl that is provided when you check the status of a complete operation. The same process may be used to view the operation metadata, available at the metadataDownloadUrl.

This is a generic HTTP GET request which is managed by the AWS S3 server, and the expected format is outlined below.

Response Elements

The asynchronous batch coding service outputs a jsonl file with each line corresponding to the record from the original input file. Each line is an AsynchronousCodeResponse object.

Examples

In response to input which specifies a record identifier:

Sample Request

GET https://domain/endpoint?queries HTTP/1.1

Sample Response

HTTP/1.1 200 OK
Date: Thu, 20 Jun 2024 02:26:34 GMT
Last-Modified: Thu, 20 Jun 2024 02:24:04 GMT
Accept-Ranges: bytes
Content-Type: application/octet-stream
Server: AmazonS3
Content-Length: 7660
...
{ "recordId": "1", “codeStatus”: “successful”, “input”: { “occp_text”: “software developer”, “tasks_text”: “writing code and unit tests” }, "result": { "codeCategory": "261313", "codeLabel": "Software Engineer", "codeConfidence": 0.98 } }
{ "recordId": "2", “codeStatus”: “unsuccessful”, “input”: { “occp_text”: “Paramedic”, “tasks_text”: “responding to medical emergencies” }, "suggestions": [{ "codeCategory": "411111", "codeLabel": "Ambulance Officer", "codeConfidence": 0.26 }, { "codeCategory": "411112", "codeLabel": "Intensive Care Ambulance Paramedic", "codeConfidence": 0.24 }]
{ "recordId": "unknown", "codeStatus": "unsuccessful", "input": "this is not a json string", "error": "Invalid JSON data format. "}

Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

In response to input which specifies no record identifier:

Sample Request

GET https://domain/endpoint?queries HTTP/1.1

Sample Response

HTTP/1.1 200 OK
Date: Thu, 20 Jun 2024 02:26:34 GMT
Last-Modified: Thu, 20 Jun 2024 02:24:04 GMT
Accept-Ranges: bytes
Content-Type: application/octet-stream
Server: AmazonS3
Content-Length: 7660
...

{ "recordId": null, “codeStatus” : “successful” , “input” : { “occp_text” : “software developer” , “tasks_text” : “writing code and unit tests” }, "result": { "codeCategory": "261313", " codeLabel": "Software Engineer", "codeConfidence": 0.98 } }
{ "id": null, “codeStatus” : “unsuccessful” , “input” : { “occp_text” : “Paramedic” , “tasks_text” : “responding to medical emergencies” }, "suggestions": [{ "codeCategory": "411111", "codeLabel": "Ambulance Officer", "codeConfidence": 0.26 }, { "codeCategory": "411112", "codeLabel": "Intensive Care Ambulance Paramedic", "codeConfidence": 0.24 }] }
{ "recordId": "unknown", "codeStatus": "unsuccessful", "input": "this is not a json string", "error": "Invalid JSON data format. "}

Reporting issues

WoAG Occupation Coding Service User Guide

Coding Service support.

Released

30/06/2025

Formatted: Font: (Default) Arial, Not Italic
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, Not Italic
Formatted: Font: (Default) Arial, Not Bold, Not Italic
Formatted: Font: (Default) Arial, Not Italic
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, Not Italic
Formatted: Font: (Default) Arial, Not Bold, Not Italic
Formatted: Font: (Default) Arial, Not Italic

If you encounter bugs or have feedback on the service, please report these via the following mechanism:

Request syntax

GET /v1/security.txt HTTP/1.1
Host: string
~~Authorisation~~Authorization: string

URI request parameters

The request does not use any URI parameters.

Request body

The request does not have a request body.

Response syntax

HTTP/1.1 200 OK
Content-type: text/html
<information on reporting errors>

Example

Download

Example

Sample request	GET /v1/security.txt HTTP/1.1 Host: https://partner-coder.api.abs.gov.au ~~Authorisation~~Authorization: example token
Sample response	HTTP/1.1 200 OK Content-type: text/html contact: mailto:security@abs.gov.au expires: 2025-08-27T05:45:00.000Z

Support for other service issues

Please contact coding.capability@abs.gov.au for other service support. Business hours are 9 am to 5 pm Monday to Friday.

Errors and suggested actions

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

WoAG Occupation Coding Service User Guide

Glossary of common errors, explanations and solutions

Released

30/06/2025

These codes help identify issues on both the client and server sides, allowing for troubleshooting and resolution of HTTP request problems.

Download

Error message	Why this happened	You should…
~~**Generic errors possible on all API calls**~~		
Invalid request body HTTP Status Code: 400	Something was wrong with your request body syntax. Example: small batch records exceed length limit of 300/you tried to code too many records at once using the synchronous small batch service.	• check the request body syntax for the API call and try again. • remove any special characters from free text inputs (see Recommended text input for coding) • If your small batch has this error, retry with a smaller batch size, or consider using the asynchronous batch coding service.
User is not authorized to access this resource with an explicit deny HTTP Status Code: 403	You have either not authenticated or your authentication token has expired. This error may also occur if there have been excessive authentication requests from others across your organisation.	• authenticate again if your token is over an hour old, or. • Review your authentication logic, and reuse your token between users and API calls (see Session token usage).

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Not Strikethrough
Commented s22]: link
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

Error message	Why this happened	You should…
~~**Generic errors possible on all API calls**~~		
Limit Exceeded HTTP Status Code: 429	You have made too many calls to the API.	• wait 24 hours before calling the API again, or • contact us to discuss increasing your request quota.
Too Many Requests HTTP Status Code: 429	You have made too many API calls in a short period of time. ~~This error may also occur if there have been excessive authentication requests from anyone else in from others across your the organisation.~~	• space out your requests over a longer timeframe, and • consider using the asynchronous coding service if coding many records.
Error performing request HTTP Status Code: 500	This happens when something unexpected goes wrong on the server. In some instances, further context is provided.	• retry after a short delay; report persistent issues to support.
Problems selecting model/topic		
The specified topic does not exist HTTP Status Code: 404	No topics matched the provided topic parameter.	• check the available topics before proceeding.
The specified model does not exist HTTP Status Code: 404	No models matched the provided model parameter. You'll see this if the system can't locate the specified model - maybe due to a typo or outdated ID.	• verify that the model name or ID is correct and still active. • check which models are available or use the default (latest) model for the topic.

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, Strikethrough
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, Bold
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

Error message	Why this happened	You should…
~~Generic errors possible on all API calls~~		
Selected model does not match the input topic. HTTP Status Code: 409	The given model GUID does not correspond to the specified coding topic.	• check which models are available or use the default (latest) model for the topic.
Errors on the get models endpoint		
No models found for topic topic HTTP Status Code: 500	No models are available for the provided topic parameter.	• reach out to your ABS contact to investigate why no model is available.
Synchronous coding errors		
Malformed record found in request HTTP Status Code: 400	The free text input did not match the expected format for the model.	• check what the expected format is for a given topic here.
Batch input contained no records HTTP Status Code: 400	You tried to code a small batch of records but the records array was empty.	• check that you have provided at least one record to be coded, and • check that your request body is correctly formatted.
There are record(s) outside the min or max char limit: Record with index x has a total text length under 3 min Record with index y has a total text length over 100 max … HTTP Status Code: 400	One or more records provided had too many or too few characters.	• Check that each record to be coded has 3-100 (inclusive) characters across the two input fields. • See Recommended text input for coding.
~~RateLimitExceededException HTTP Status Code: 429 (Too Many Requests)~~	~~You'll see this when sending too many requests too quickly.~~	• ~~space out your requests and implement retry logic.i~~

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, Bold
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, Bold
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

Error message	Why this happened	You should…
~~Generic errors possible on all API calls~~		
	~~This error may also occur if there have been excessive authentication requests from anyone else in the organisation.~~	
Asynchronous coding errors		
User is not authorised to retrieve operation GUID HTTP Status Code: 401	The specified operation does not belong to the current user. You may have authenticated with the wrong user or specified the wrong operation_id.	• authenticate again and check you have the right credentials, and • confirm your operation id.
Unable to retrieve operation for given id HTTP Status Code: 404	No operations were found to match the given operation_id. This is most likely due to a typo.	• confirm the operation ID is correct. • if you have lost your operation ID, you will have to create a new operation.

See more information on HTTP errors at HTTP response status codes - HTTP | MDN.

Glossary of inputs and responses

WoAG Occupation Coding Service User Guide

Model details, Record and Response objects.

Released

30/06/2025

Model details

Download

These fields are returned by various methods in Gathering parameters:

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, Bold
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

Model details

Field	Description	Type
modelId	Unique identifier used to reference the ML model.	String (GUID format)
modelVersion	Version number of the model trained on the topic. Distinct from topicVersion.	Number
modelReleaseDate	Date the model was released, in ISO8601 format.	String
modelType	ML algorithm used (e.g., hsvm).	String
inputFormat	List of expected input field names.	Array of strings
topicStandard	Full name of the classification topic.	String
topicVersion	Version number of the topic classification used in model training.	String

RecordObject

Download

These are the fields expected when submitting data to the coding service:

RecordObject

Field	Description	Type
occp_text	Free-text description of an occupation.	String
tasks_text	Tasks or duties related to the occupation.	String
recordId	Optional identifier for each input record.	String

Response objects

These are returned after synchronous or asynchronous coding operations:

SynchronousCodeResponse

Download

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial

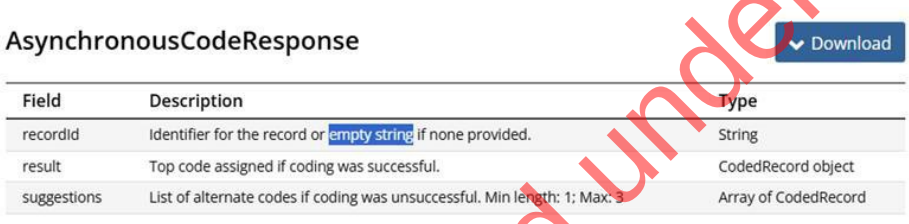
SynchronousCodeResponse

Field	Description	Type
recordId	Identifier for the submitted input record (if originally provided).	String
codeStatus	Coding outcome. Valid values: successful, unsuccessful.	String
input	Input record submitted to the model.	RecordObject
result	List of predicted codes and labels. Min length: 0; Max: 16 (or value of numberOfSuggestions)	Array of CodedRecord

AsynchronousCodeResponse

Download

AsynchronousCodeResponse



Field	Description	Type
recordId	Identifier for the record or ~~empty string~~null if none provided.	String
codeStatus	Coding outcome. Valid values: successful, unsuccessful.	String
input	Input record submitted to the model.	RecordObject
result	Top code assigned if coding was successful.	CodedRecord object
suggestions	List of alternate codes if coding was unsuccessful. Min length: 1; Max: 3	Array of CodedRecord

CodedRecord

Formatted: Font: (Default) Arial

Download

CodedRecord

Field	Description	Type
codeCategory	Code assigned to the input.	String
codeLabel	Description of the code category.	String
codeConfidence	Confidence score (e.g., 0.92). May be rounded or multiplied by 100 for a percentage.	

Coding Service security

WoAG Occupation Coding Service User Guide

Coding Service system security controls.

Released

30/06/2025

The WoAG Occupation Coding Service and API has been security assessed by an independent registered assessor within the Australian Signals Directorate (ASD) Information Security Registered Assessors Program (IRAP) Program. This assessment found the Coding Service and API to have met the control and security objectives defined through the Australian Government Information Security Manual (ISM).

Agencies may need to sign off in-house on using an external API, for business, legal, or security reasons. They may also need to check on their own behalf that the API response is from the address they sent the request to.

The following security controls, drawn from the ISM, are included to assist partner agencies in assessing their risks when using this service.

Download

Control name	System security controls
Cryptography	• Data is encrypted in transit to and from the API. All APIs created with Amazon API Gateway expose HTTPS endpoints only. API Gateway does not support unencrypted (HTTP) endpoints.

Control name	System security controls
	• API Gateway has been configured to choose a minimum Transport Layer Security (TLS) protocol version of TLS 1.2. • Data is encrypted at rest if it is to be stored by the request action; only file-based batch requests require the storage of request data. KMS Keys will be created for each environment and applied to data (or metadata) storage components e.g., S3 buckets, DynamoDB tables.
Data transfers	• Bulk data transfers only occur for asynchronous requests. This occurs through the AWS WAF and the AWS ABS Gateway. • Data transferred as part of an asynchronous request is scanned. • Resources involved in the coding of file-based requests (e.g. Lambda, SageMaker) are able to read data from bucket(s) containing post-scanned/validated data, not raw ingress data, and are only able to write to a specific egress bucket. • For file-based batch coding, consumers are provided S3 pre-signed URLs for data file ingress and egress. This aligns with ISM cryptographic control requirements (ISMF 1123–1126), access control measures for external interfaces (ISMF 1295–1300), and secure transmission/storage of sensitive data (ISMF 1352–1354). • These URLs are configured with expiration times aligned with the time required to perform a coding request. This is a dynamic value and is configured based on performance data from implemented models. • Data stored in service of file-based batch coding (e.g. ingressed request data and egressed coded response data) is configured for automated expiry (deletion). Minimal expiry time is configured based on the time required to perform a coding request, pending performance data from implemented models.
Data sovereignty	• No data will be stored or processed outside Australia. • Services will never failover to services outside of Australia.

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial

Control name	System security controls
Machine Learning (ML)	• There is no external connection (outside of the dedicated ABS accounts) or other ML reference used in the WoAG Coding Service or in the training of the ML models. • Only isolated instances of Machine Learning within ABS-owned secure AWS accounts are used to train the models that underpin the Coding Service. • While the ABS is exploring the use of Distilbert - Large Language Models (LLM) combined with Census data to train coder models, only more traditional ML models such as Hierarchical Support Vector Machine (HSVM) models, trained only using Census data, will be used in the external service. • Only specific response text, separated from all other response data, is used to create the ML models and in the Coding Service. • The service can only respond with the classification codes and labels defined in the relevant classification standard and version, unless a record identifier is also provided by the user. In this instance, the record identifier is returned to the user with the data. • The application of the models used in the Coding Service, and all data passed through the coders, remains within the ABS secure accounts at all times. No user data is stored or retained. User data is temporarily stored within ABS secure accounts while being processed, and then deleted.

Additional resources

The following document provides example scripts to assist agencies. The ABS does not provide maintenance or support for end users' own applications, but the examples may provide a useful starting point for integrating with the coding API, or converting batch coding output.

Integration script examples.docx

Formatted: Font: (Default) Arial

Field Code Changed

Formatted: Font: (Default) Arial

Coding API Integration Examples

The ABS does not provide maintenance or support for end users' own applications, but the following examples may provide a useful starting point for integrating with the coding API, or converting batch coding output.

The scripts and code snippets were written as a proof of concept and may not scale to large datasets.

Consider using a streaming approach if you encounter performance issues.

Contents

1. Manager's guide: technical integration requirements

The coding service is designed for formal integration and requires the expertise of a qualified technical team. This team should have the capability to call APIs and manage integrations with your agency's internal data systems and workflow processes.

Your technical team will need to refer to the Coding Service User Guide to develop a custom integration that aligns with your organisation's development standards. This may include the creation of automated workflows or user interfaces to enhance usability and efficiency.

To support your initial evaluation, we have provided a set of sample scripts. Please note that these are intended for demonstration purposes only – they are not production-ready and are not supported by our team.

Suggested workflow

Step 1: Initial Testing

- Managers should share the PowerShell starter scripts below with their technical teams. These scripts are designed to help assess suitability with your organisation's systems and needs.
- ~~Ensure that the scripts can~~ These example scripts may be able to run in your organisation without modifications.
- If adjustments are required, your technical team should modify the scripts to suit your infrastructure. The ABS cannot support such customisations as they are highly user dependent.

Step 2: Evaluation and Planning

- Evaluate the results of using the service.
- Plan out the future integration and workflow that will meet your business needs.

Step 3: Implementation and Testing

- Refer to the Coding Service User Guide for technical details.
- Collaborate with your technical teams to implement and test the integration.
- For questions or clarification, contact coding.capability@abs.gov.au.

Step 4: Ongoing support and ownership

- Your organisation is responsible for its own integrations, supported by your internal technical teams.
- If issues arise that cannot be resolved internally, these may be escalated to the ABS for further assistance.

Steps 2-4 are expected to be managed through your organisation's internal workflows. The remainder of this guide focuses on supporting **Step 1 – the initial testing phase**.

Once the integration is established, your technical teams will be best positioned to provide clear instructions for end users on how to operate and maintain the solution.

Initial testing considerations

When setting up the initial integration, your technical team will need to consider the following potential issues:

- Use of the coding service does not comply with your organisation's existing data governance policies.
    - You may need to obtain internal approvals to use the service. For more information on how the coding service handles user data, see Coding Service Security.
    - There may be technical restrictions which prevent you from using the coding service. These can likely be modified once you have the required approvals.
- Access to the API is blocked by the organisation's proxy server.
    - Your organisation may use a web proxy to improve web performance, protect users' privacy and restrict unapproved web content.
    - Your integration may need to be modified to accommodate the proxy, or you may need to review your proxy settings to allow access to the coding API.
- Access to the API is blocked by the organisation's firewall.
    - A firewall may be used to block malicious traffic from entering your organisation's network.
    - Depending on how your firewall is configured, your technical team may need to explicitly allowlist access to the coding API.
- You cannot run PowerShell scripts in your organisation.
    - Your technical team may need to develop a custom implementation in line with your organisation's software development processes. Refer to the Coding Service User Guide for technical details.

2. Authentication

The following PowerShell script will:

1. construct an encoded authentication string from your provided client ID and client secret,
2. fetch a temporary authentication token from the AWS Cognito token issuer endpoint, and
3. set up the headers you will need for subsequent API calls.

You need to have an active authentication token to make any API calls. Authentication tokens last for one hour and can be reused between API calls and between users in an organisation.

You will need to set the following variables:

1. Cognito hostname: this will depend on whether you are registered as a public or partner user:
   - Public users: “https://public-coder.auth.abs.gov.au”
   - Partner users: “https://partner-coder.auth.abs.gov.au”
2. Client ID: provided to you upon registration.
3. Client secret: provided to you upon registration.

```
$cognitoHostname = "COGNITO_HOSTNAME";
$clientId = "CLIENT_ID";
$clientSecret = "CLIENT_SECRET";

# Construct authorisation token (base64 version of <clientId:clientSecret>)
$authorisationToken =
[Convert]::ToBase64String([char[]]"$($clientId):$($clientSecret)");

# Call Cognito to get the access token
$headers = @{
'Authorization' = "Basic $($authorisationToken)"
'Content-Type' = "application/x-www-form-urlencoded"
};
$body = @{
grant_type = "client_credentials"
};
$cognitoResponse = Invoke-RestMethod -Method 'Post' -Uri
"$($cognitoHostname)/oauth2/token" -Headers $headers -Body $body;
$authorisationToken = $cognitoResponse.access_token

# Set headers for coding API calls
$headers = @{
'Authorization' = "$authorisationToken"
'Content-Type' = "application/json"
};
```

3. Gathering Parameters

These PowerShell examples only cover the use of the latest (default) model for each occupation classification, and up to three codes per record. To do this, set the following variables:

- API hostname: this will also depend on whether you are registered as a public or partner user.
    - Public users: “https://public-coder.api.abs.gov.au”
    - Partner users: “https://partner-coder.api.abs.gov.au”
- Coding topic: this is the short form of the classification you are coding against.
    - Australian and New Zealand Standard Classification of Occupations: “anzsco”.
    - Occupation Standard Classification for Australia: “osca”.

```
$hostname = "API_HOSTNAME";
$codingTopic = "CODING_TOPIC";
```

4. Real-time (synchronous) coding

The input for synchronous coding will depend on whether you are coding a single record or a small batch of records. One PowerShell example of each scenario is shown below.

Single Record

```
$body = @{
    'record' = @{
        occp_text = "software developer"
        tasks_text = "writing code and unit tests"
    }
    'numberOfSuggestions' = 3
} | ConvertTo-Json
```

Small Batch of Records

```
$body = @{
    'records' = (
        @{
            occp_text = "software developer"
            tasks_text = "writing code and unit tests"
        },
        @{
            occp_text = "Paramedic"
            tasks_text = "respond to emergencies"
        },
        @{
            recordId = "1"
            occp_text = "Sales assistant"
        }
    )
    'numberOfSuggestions' = 3
} | ConvertTo-Json
```

Coding your input

Once you have set up your input body, run the following script to code it against the API and save the coded response.

```
$apiResponse = Invoke-RestMethod -Method 'Post' -Uri
"$hostname/topics/$codingTopic/code" -Headers $headers -Body $body
```

Reading the coded output

You can then output the coded response in any of the following formats. See the bolded sections to edit the output file name.

JSON output

PowerShell Script:

```
$apiResponse | ConvertTo-Json > "output.json"
```

Example output (single code):

```
{
    "codeStatus":  "successful",
    "input": {
        "tasks_text":  "writing code and unit tests",
        "occp_text":  "software developer"
    },
    "result": [
        {
            "codeCategory":  "261313",
            "codeLabel":  "Software Engineer",
            "codeConfidence":  0.61
        }, {
            "codeCategory":  "261312",
            "codeLabel":  "Developer Programmer",
            "codeConfidence":  0.44
        }, {
            "codeCategory":  "261399",
            "codeLabel":  "Software and Applications Programmer nec",
            "codeConfidence":  0.44
        }
    ]
}
```

Example output (small batch code):

```
[
    {
        "codeStatus": "successful",
        "input": {
           "occp_text": "software developer",
            "tasks_text": "writing code and unit tests"
        },
        "result": [{
            "codeCategory": "261313",
            "codeLabel": "Software Engineer",
            "codeConfidence": 0.61
```

```
        },{
            "codeCategory": "261312",
            "codeLabel": "Developer Programmer",
            "codeConfidence": 0.44
        },{
            "codeCategory": "261399",
            "codeLabel": "Software and Applications Programmers",
            "codeConfidence": 0.44
        }]
    }, {
        "codeStatus": "unsuccessful",
        "input": {
            "occp_text": "Paramedic, respond to emergencies"
        },
        "result": [ ]
    }, {
        "recordId": "1",
        "codeStatus": "unsuccessful",
        "input": {
            "occp_text": "Sales assistant"
        },
        "result": [ ]
    }
]
```

CSV output

PowerShell Script:

```
$apiResponse | ForEach-Object {
  $normalised = [ordered] @{}
  # Top level keys
  foreach ($key in @("recordId", "codeStatus")) {
    if ($_.PSObject.Properties[$key]) {
      $normalised[$key] = $_.$key
    } else {
      $normalised[$key] = $null
    }
  }
  # Expand input object
  foreach ($key in @("occp_text", "tasks_text")) {
    if ($_.PSObject.Properties["input"] -And
$_.input.PSObject.Properties[$key]) {
      $normalised["input.$key"] = $_.input.$key
    } else {
      $normalised["input.$key"] = $null
    }
  }
  # Expand result array
  for ($i=1; $i -le 3; $i++) {
    foreach ($key in @("codeCategory", "codeLabel", "codeConfidence")) {
      if ($_.PSObject.Properties["result"] -And $_.result.Count -ge $i -
And $_.result[$i-1].PSObject.Properties[$key]) {
        $normalised["result_$i.$key"] = $_.result[$i-1].$key
      } else {
        $normalised["result_$i.$key"] = $null
      }
    }
  }
  [PSCustomObject]$normalised
} | Export-Csv -Path "example.csv" -NoTypeInformation
```

Example output (single code):

recordId	codeStatus	input .occp_text	input .tasks_text	result_1 .codeCategory	result_1 .codeLabel	result_1 .codeConfidence
	successful	software developer	writing code and unit tests	261313	Software Engineer	0.61

(columns continued)

result_2 .codeCategory	result_2 .codeLabel	result_2 .codeConfidence	result_3 .codeCategory	result_3 .codeLabel	result_3 .codeConfidence
261312	Developer Programmer	0.44	261399	Software and Applications Programmers nec	0.44

Example output (small batch code):

recordId	codeStatus	input .occp_text	input .tasks_text	result_1 .codeCategory	result_1 .codeLabel	result_1 .codeConfidence
	successful	software developer	writing code and unit tests	261313	Software Engineer	0.61
	unsuccessful	Paramedic, respond to emergencies				
1	unsuccessful	Sales assistant				

(columns continued)

result_2 .codeCategory	result_2 .codeLabel	result_2 .codeConfidence	result_3 .codeCategory	result_3 .codeLabel	result_3 .codeConfidence
261312	Developer Programmer	0.44	261399	Software and Applications Programmers nec	0.44

5. Asynchronous batch coding

The process of asynchronous coding involves several steps, shown in the following sample PowerShell scripts:

1. Create a new batch coding operation

```
$apiResponse = Invoke-RestMethod -Method 'Post' -Uri
"$hostname/topics/$codingTopic/batch-code" -Headers $headers -Body '{}'
```

2. Upload your input file (located at *filename.jsonl*)

```
$uri = $apiResponse.requestUploadUrl
$operationId = $apiResponse.operationId
# upload input file
$putHeaders = @{
    'Content-Type'="application/json"
    'x-amz-server-side-encryption'="aws:kms"
    'x-amz-server-side-encryption-aws-kms-key-
id'="$($apiResponse.bucketKmsKeyArn)"
};
$putResponse = Invoke-RestMethod -Method 'Put' -Uri
"$($apiResponse.requestUploadUrl)" -Headers $putHeaders -InFile
"filename.jsonl"
```

3. Check your operation status every 10-15 minutes, until you get the status “complete”

```
$apiResponse = Invoke-RestMethod -Method 'Get' -Uri
"$hostname/topics/$codingTopic/batch-code/operations/$operationId" -Headers
$headers
Write-Host ($apiResponse | Format-List | Out-String)
```

4. Download the output file to *filename.jsonl*

```
Invoke-RestMethod -Method 'Get' -Uri $apiResponse.responseDownloadUrl >
"filename.jsonl"
```

6. Converting batch coding output

The asynchronous batch coding service generates a JSONL file containing the results for each input record. This is a standard format for internet communication and can be easily converted to other common data formats as part of an integrated application. One such format is CSV, which can be opened in Microsoft Excel.

Example output from coding service (JSONL)


```
{ "recordId": "1", "codeStatus": "successful", "input": { "occp_text":
"software developer", "tasks_text": "writing code and unit tests" },
"result": { "codeCategory": "261313", "codeLabel": "Software Engineer",
"codeConfidence": 0.98 } }
{ "recordId": "2", "codeStatus": "unsuccessful", "input": { "occp_text":
"Paramedic", "tasks_text": "responding to medical emergencies" },
"suggestions": [ { "codeCategory": "411111", "codeLabel": "Ambulance
Officer", "codeConfidence": 0.26 }, { "codeCategory": "411112",
"codeLabel": "Intensive Care Ambulance Paramedic", "codeConfidence": 0.24
} ] }
{ "recordId": "unknown", "codeStatus": "unsuccessful", "input": "this is
not json", "error": "invalid JSON" }
```


Converted CSV as shown in a spreadsheet

recordId	codeStatus	error	input.occp_text	input .tasks_text	result .codeCategory	result .codeLabel
1	successful		software developer	writing code and unit tests	261313	Software Engineer
2	unsuccessful		Paramedic	responding to medical emergencies		
unknown	unsuccessful	invalid JSON				

(columns continued)

result .codeConfidence	suggestions_1 .codeCategory	suggestions_1 .codeLabel	suggestions_1 .codeConfidence	suggestions_2 .codeCategory
0.98				Software Engineer
	411111	Ambulance Officer	0.26	

(columns continued)

suggestions_2 .codeLabel	suggestions_2 .codeConfidence	suggestions_3 .codeCategory	suggestions_3 .codeLabel	suggestions_3 .codeConfidence
	411111	Ambulance Officer	0.26	

Converting to .csv - sample PowerShell script

Remember to replace the file names with your own files.

```
$rawObjects = Get-Content "example.jsonl" | ForEach-Object { $_ | ConvertFrom-
JSON }
$normalisedObjects = $rawObjects | ForEach-Object {
  $normalised = [ordered] @{}
  # Top level keys
  foreach ($key in @("recordId", "codeStatus", "error")) {
    if ($_.PSObject.Properties[$key]) {
      $normalised[$key] = $_.$key
    } else {
      $normalised[$key] = $null
    }
  }
  # Expand input object
  foreach ($key in @("occp_text", "tasks_text")) {
    if ($_.PSObject.Properties["input"] -And
$_.input.PSObject.Properties[$key]) {
      $normalised["input.$key"] = $_.input.$key
    } else {
      $normalised["input.$key"] = $null
    }
  }
  # Expand result object
  $codeFields = @("codeCategory", "codeLabel", "codeConfidence")
  foreach ($key in $codeFields) {
    if ($_.PSObject.Properties["result"] -And
$_.result.PSObject.Properties[$key]) {
      $normalised["result.$key"] = $_.result.$key
    } else {
      $normalised["result.$key"] = $null
    }
  }
  # Expand suggestions array
  for ($i=1; $i -le 3; $i++) {
    foreach ($key in $codeFields) {
      if ($_.PSObject.Properties["suggestions"] -And $_.suggestions.Count -ge
$i -And $_.suggestions[$i-1].PSObject.Properties[$key]) {
        $normalised["suggestions_$i.$key"] = $_.suggestions[$i-1].$key
      } else {
        $normalised["suggestions_$i.$key"] = $null
      }
    }
  }
  [PSCustomObject]$normalised
}
$normalisedObjects | Export-Csv -Path "example.csv" -NoTypeInformation
```

Converting to .csv - sample JavaScript code

Remember to replace the file names with your own files.

```
import { readFileSync, writeFileSync } from "fs";
import { stringify } from "csv-stringify/sync";

// Read code results from API
const data = readFileSync("example.jsonl", "utf-8")
  .split("\n").filter(Boolean).map(JSON.parse);

// Gather column names and flatten JSON objects
const columns = ["recordId", "codeStatus", "error"];
for (let inputField of ["occp_text", "tasks_text"]) {
  columns.push(`input.${inputField}`);
  data.map((record) => { record[`input.${inputField}`] = record.input
      ? record.input[inputField] : null;
  });
}
const codeFields = ["codeCategory", "codeLabel", "codeConfidence"];
for (let codeField of codeFields) {
  columns.push(`result.${codeField}`);
  data.map((record) => { record[`result.${codeField}`] = record.result
      ? record.result[codeField] : null;
  });
}
for (let i = 0; i < 3; i++) {
  for (let codeField of codeFields) {
    columns.push(`suggestions_${i + 1}.${codeField}`);
    data.map((record) => { record[`suggestions_${i + 1}.${codeField}`] =
        record.suggestions && record.suggestions[i]
          ? record.suggestions[i][codeField] : null;
    });
  }
}

// Output to CSV file
writeFileSync(
  "example.csv",
  stringify(data, {
    header: true,
    columns: columns,
  }),
);
```

Converting to .csv - sample bash script (Linux command line interface)

Remember to replace the file names with your own files. This script requires jq to be installed.

```
export set inputFile="example.jsonl"
export set outputFile="example.csv"
echo
"recordId,codeStatus,error,input.occp_text,input.tasks_text,result.codeC
ategory,result.codeLabel,result.codeConfidence,suggestions_1.codeCategor
y,suggestions_1.codeLabel,suggestions_1.codeConfidence,suggestions_2.cod
eCategory,suggestions_2.codeLabel,suggestions_2.codeConfidence,suggestio
ns_3.codeCategory,suggestions_3.codeLabel,suggestions_3.codeConfidence"
> $outputFile
jq -s '.' $inputFile | jq -r '
  .[] |
  [
    .recordId // "",
    .codeStatus // "",
    .error // "",
    .input.occp_text // "",
    .input.tasks_text // "",
    .result.codeCategory // "",
    .result.codeLabel // "",
    .result.codeConfidence // "",
    .suggestions[0].codeCategory // "",
    .suggestions[0].codeLabel // "",
    .suggestions[0].codeConfidence // "",
    .suggestions[1].codeCategory // "",
    .suggestions[1].codeLabel // "",
    .suggestions[1].codeConfidence // "",
    .suggestions[2].codeCategory // "",
    .suggestions[2].codeLabel // "",
    .suggestions[2].codeConfidence // ""
  ] | @csv
' >> $outputFile
```

Converting to .csv - sample Python script

Remember to replace the file names with your own files.

```
import pandas as pd

# Read the dictionary into a data frame and add a common prefix
def dict_to_df(data: dict, column_prefix: str):
   return pd.DataFrame([data]).add_prefix(column_prefix)

# Expand each dictionary in a list into its own set of columns
def format_list(row, column_prefix):
  if not isinstance(row, list) or not row:
      row = [{}]
  # Concatenate custom data frames for each item in the list
  return pd.concat([dict_to_df(list_item, f'{column_prefix}_{i+1}.') for
i, list_item in enumerate(row)], axis=1)

df = pd.read_json("example.jsonl", orient="records", lines=True)

# Expand nested JSON objects
input_df = pd.json_normalize(df["input"]).add_prefix("input.")
result_df = pd.json_normalize(df["result"]).add_prefix("result.")
df = pd.concat([df.drop(columns=["input", "result"]), input_df,
result_df], axis=1)

# Expand suggestions list
suggestion_df = df['suggestions'].apply(lambda suggestions:
format_list(suggestions, "suggestions"))
suggestion_df = pd.concat(suggestion_df.to_list(), ignore_index=True)
suggestion_df.index = df.index

# merge the suggestions to the original data
merged_df = df.join(suggestion_df, how="left")
merged_df = merged_df.drop('suggestions', axis=1)

merged_df.to_csv("example.csv", index=False)
```