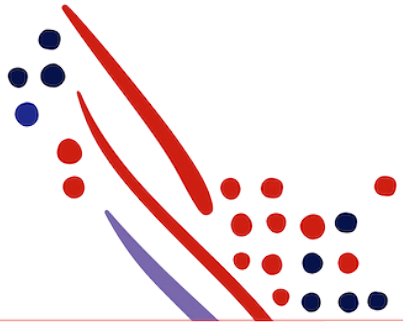
Guide

Data Collection Entries API Guide for Workforce Now Tempus PI

Published on
May 16, 2024, 03:28 PM

Last modified
May 16, 2024, 03:38 PM

Always Designing for People™



ADP Copyright Information

Published on
May 16, 2024, 03:28 PM

Published on
May 16, 2024, 03:38 PM

Table of Contents

Chapter 4

Message Code Dictionary – userMessage.CodeValue

Errors

Chapter 5

Frequently asked questions (FAQs):

Question 1: What if the clock isn't syncing when you are trying to do punches within 2 mins.Meaning as an example,clock in at 11:10 AM EST and clock out at 11:12 AM EST ?

Chapter 1

Overview

The API is used to submit punches for an employee.

Below is a summary of the web services documented in this API:

Service	HTTP Verb	Description
data-collection-entries.process()	POST	Used for recording clocking transactions such as “Clock-In,” “Clock-Out,” “Scan" etc
data-collection-entries.process()/{eventid}	GET	Used to report the status of a previously submitted upload.

- Each service returns a JSON object in its response.
- Services using the POST HTTP method (verb) will expect a JSON object in its request body.

What's New in this Guide?

Added new section Multiple Punches Sample,this has sample payload to upload multiple punches for multiple employees in one request

January 2024

Added new FAQ on clock out punches within 2 mins.

Postman Collection

Postman allows you to import a collection of APIs, created by others, so you can try them out. For more information on Postman, see Making Your First API Call Using Postman.

To download API collections for the Team Time Cards API from the ADP GitHub library and import them to Postman, go to Data Collection Entries TLM API Postman Collection.

High Level Interaction Diagram.

Following diagram depicts interaction on data-collection-entries API call once consumer is authenticated by ADP market place.

Supported Product Version and Customer Base

WFN Tempus 24.0.0.0 and above

Required Setup Steps

Employee has to be valid payroll cycle with Clocking time entry plan

Chapter 2

Data Collection Entries Web Service Method - Upload - POST

The data-collection-entries.process HTTP POST method is used to upload a payload of punches to the ADP TLM servers for processing.

Action Code/ Entry Codes Overview

This section describes the validate Action code and Entry Code to be used in the API.

Action and Entry Code JSON Object Locations

Element Name	JSON Object Location
Action Code	/events/data/transform/dataCollectionEntries/workerDataCollectionEntries/actionCode/codeValue
Entry Code	/events/data/transform/dataCollectionEntries/workerDataCollectionEntries/entryCode/codeValue

Action Code/Entry Code Description Matrix

Action Code	Entry Code	Description
punch	NA	Scan operation (first scan represents an "IN"; second scan represents an "OUT").
lunchout	NA	Clock out for lunch operation.
clockin	NA	Clock-in operation
clockout	NA	Clock-out operation

Action Code/Entry Code Not Supported for Tempus PI

Action Code	Entry Code	Description
Transfer	**NA**	**Due to other dependencies transfer functionality is not supported to marketplace clients for TEMPUS PI**
earnings	**{required}**	**Earning are not supported for TEMPUS PI**
suppEarnings	**{required}**	**Earning are not supported for TEMPUS PI**

General Comments and Notes

- The target clocking employees are always specified in the body by using the Badge value
- For all HTTP Status responses of 202, The "Location" and "Retry-After" response headers indicate that a subsequent request can be made to the data-collection-entries/{eventid} HTTP GET method to obtain an import and/or timecard posting status of the uploaded punches
- The data-collection-entries.process web service is designed to be used asynchronously:
    - Honoring the "Retry-After" response header returned by the data-collection-entries.process and data-collection-entries/{eventid} web service pertains only to the subsequent requests to obtain a status for a particular Event ID (using data-collection-entries/{eventid}).
    - Subsequent requests can be made to the data-collection-entries.process web service to upload additional punches while asynchronously polling/waiting for previous bulk upload events to complete
    - To limit inadvertent DOS (Deinial of Service) incidences, ADP asks consuming applications to honor a wait interval of at least one minute between each subsequent bulk upload request made to data-collection-entries.process.

- Refer to the latest "ADP APIs - Data Dictionary.pdf" document for the following:
    - Creating your Personal Information Exchange (PFX) Certificate.
    - Installing Your PFX Certificate.
    - Installing and Preparing Google Postman
    - Requesting a Bearer Token

Request Header Parameters

Header Name	Sample value	Comments
Authorization	Bearer {accessToken}	The Token Type (accessToken—i.e. "Bearer") plus the Access Token (access_token—i.e. "edc3c2-6b5d-4251-b258-d1208cbf6339") returned from ADP Marketplace's Token web service.

Multiple Punches Sample

This is an example of a request that contains multiple time punch entries in the uploading payload

API:	/events/time/v1/data-collection-entries.process
Method:	POST
Condition:	The Payload has been accepted and stored for further processing by ADP Marketplace. Subsequent requests to the datacollection entries.process, Status service are required by providing the **eventID** value returned in the **confirmMessage** object within the response to confirm the processing status of the submitted bulk upload. The Response Code applies specifically to the POST Method request.
GitHub Sample Request/Sample Response Payload:	**Request:** Sample punch.json **Response:** response.json

Scan/Punch

Performs a scan punch operation where the first scan represents an "IN" punch and the next scan represents an "OUT" punch.

API:	/events/time/v1/data-collection-entries.process
Method:	POST
Condition:	The Payload has been accepted and stored for further processing by ADP Marketplace. Subsequent requests to the datacollection entries.process, Status service are required by providing the **eventID** value returned in the **confirmMessage** object within the response to confirm the processing status of the submitted bulk upload. The Response Code applies specifically to the POST Method request.
GitHub Sample Request/Sample Response Payload:	**Request:** punch request.json **Response:**

	response.json

Lunch Out

Performs a lunch out operation.

API:	/events/time/v1/data-collection-entries.process
Method:	POST
Condition:	The Payload has been accepted and stored for further processing by ADP Marketplace. Subsequent requests to the datacollection entries.process, Status service are required by providing the **eventID** value returned in the **confirmMessage** object within the response to confirm the processing status of the submitted bulk upload. The Response Code applies specifically to the POST Method request.
GitHub Sample Request/Sample Response Payload:	**Request:** punch request.json **Response:** response.json

Clock In

Performs a clock-in operation.

API:	/events/time/v1/data-collection-entries.process
Method:	POST
Condition:	The Payload has been accepted and stored for further processing by ADP Marketplace. Subsequent requests to the datacollection entries.process, Status service are required by providing the **eventID** value returned in the **confirmMessage** object within the response to confirm the processing status of the submitted bulk upload. The Response Code applies specifically to the POST Method request.
GitHub Sample Request/Sample Response Payload:	**Request:** clockin.json **Response:** response.json

Clock Out

API:	/events/time/v1/data-collection-entries.process
Method:	POST
Condition:	The Payload has been accepted and stored for further processing by ADP Marketplace. Subsequent requests to the datacollection entries.process, Status service are required by providing the **eventID** value returned in the **confirmMessage** object within the response to confirm the processing status of the submitted bulk upload.

	The Response Code applies specifically to the POST Method request.
GitHub Sample Request/Sample Response Payload:	**Request:** **Response:** response.json

Sample Supervisor Edit

API:	/events/time/v1/data-collection-entries.process
Method:	POST
Condition:	The Payload has been accepted and stored for further processing by ADP Marketplace. Subsequent requests to the datacollection entries.process, Status service are required by providing the **eventID** value returned in the **confirmMessage** object within the response to confirm the processing status of the submitted bulk upload. The Response Code applies specifically to the POST Method request.
GitHub Sample Request/Sample Response Payload:	**Request:** supervisor edit dit.json **Response:** response.json

Alternate Worker ID Samples

Sometimes a consumer may want to identify the Employee Position for the time punch being uploaded using another commonly known attribute such as an ASSOCIATEOID rather than using badge.

Using PositionID

API:	/events/time/v1/data-collection-entries.process
Method:	POST
Condition:	The Payload has been accepted and stored for further processing by ADP Marketplace. Subsequent requests to the datacollection entries.process, Status service are required by providing the **eventID** value returned in the **confirmMessage** object within the response to confirm the processing status of the submitted bulk upload. The Response Code applies specifically to the POST Method request.
GitHub Sample Request/Sample Response Payload:	**Request:** positionid.json **Response:** response.json

Using ASSOCIATEOID

Below is a sample of the ASSOCIATEOID being used to identify the Employee Position that is associated with the Time Punch.

API:	/events/time/v1/data-collection-entries.process
Method:	POST
Condition:	The Payload has been accepted and stored for further processing by ADP Marketplace. Subsequent requests to the datacollection entries.process, Status service are required by providing the **eventID** value returned in the **confirmMessage** object within the response to confirm the processing status of the submitted bulk upload. The Response Code applies specifically to the POST Method request.
GitHub Sample Request/Sample Response Payload:	**Request:** Associateoid.json **Response:** response.json

Using Work Assignment ID (PFID)

Below is a sample of the WORKASSIGNMENTID (PFID) being used to identify the Employee Position that is associated with the Time Punch.

API:	/events/time/v1/data-collection-entries.process
Method:	POST
Condition:	The Payload has been accepted and stored for further processing by ADP Marketplace. Subsequent requests to the datacollection entries.process, Status service are required by providing the **eventID** value returned in the **confirmMessage** object within the response to confirm the processing status of the submitted bulk upload. The Response Code applies specifically to the POST Method request.
GitHub Sample Request/Sample Response Payload:	**Request:** workassignmentid.json **Response:** response.json

Data Collection Entries API Element Glossary

Selected important elements of the payload API are defined below for user reference and information. Any elements not documented below are assumed to be non-applicable and can be ignored for purposes of the data-collection-entries HTTP POST web service method. Elements that have a "Required?" value of "N/A" are data elements that are typically found in the response of an upload request (and would not be needed in the upload request itself but are documented to indicate their importance in subsequent requests that might be made in order to obtain an upload status).

Events[]

Element Name	Description	Required?
events[]	An array of events. Typically, only one event (events[0]) need ever be supplied. The events[] array element is the parent object of the dataCollectionEntries[] array element.	Yes

Element Name	Description	Required?
events[].eventID	The eventID is the value that can be used to reference an upload event in a subsequent request to obtain upload status. It typically is found in the response API of the initial data-collection-entries.process upload request made.	N/A
events[].serviceCatogory Code.codeValue	Should always have a value of "core" (constant value).	Yes

dataCollectionEntries[]

The table below describes the elements of the events[].data.transform.dataCollectionEntries[] array objects.

Element Name	Description	Required?
itemID	The itemID is used to identify the dataCollectionEntries array element. It is typically an integer value that represents the indexed position of the array element being described. For example, dataCollectionEntries[0] would have a value of "0", events[1].eventID would have a value of 1, etc.	Yes
terminalNam e	Represents the ID of the terminal associated with the punches described by the workerDataCollectionEntries[] array (clock name).	NO

workerDataCollectionEntries[]

The table below describes selected elements of the events[].data.transform.dataCollectionEntries[].workerDataCollectionEntries array objects. All other elements of the workerDataCollectionEntries array objects not defined here can be ignored and are not required.

Element Name	Description	Required?
Element Name	Description	Required?
entryID	The entryID is used to identify the workerDataCollectionEntries array element. It is typically an integer value that represents the indexed position of the array element being described. For example, dataCollectionEntries[0] would have a value of "0", events[1].eventID would have a value of 1, etc.	Yes
entryCode.cod eValue	Used to assign the Earning or Supplemental earning code assocated with an "earnings" or "suppEarnings" action code. This is a required value whenever an "earnings" or "suppEarnings" action code is specified. Not required for all other cases. See "Standard ADP Available Entry Code Values" for more information about the codes that can be used for this assignment.	Required when used with the "earnings" and "suppEarnings" action codes
badgeID	The badge value associated with the time punch being recorded.	Yes
actorBadgeID	The badge value associated with a "punch edit" made at the clock. Typically, it represents the badge value of the supervisor adding a missed punch or recording an allocation on behalf of an employee.	No

Element Name	Description	Required?
deviceDateTime	The timestamp of the data collection device for the punch being recorded. It has the following format: YYYY-MM-DDTHH:MI:SS-HH:MI For example: 2016-11-11T11:30:15-05:00 Where "-05:00" is the hour/minutes offset from UTC.	
entryDateTime	The timestamp of the Punch being recorded for the employee. When the punch is not an edit (either a Supervisor or Employee Edit operation at the clock), the entryDateTime will typically match the deviceDateTime. Uses the same date format as deviceDateTime.	Yes
actionCode	The action code associated with the punch being recorded. See "Action Code/Entry Code Description Matrix" for the set of action codes that can be used for this assignment.	Yes
alternateWokerIDs	Used to identify the Employee Position belonging to the punch by alterantive elements, such as ASSOCIATEOID. To date, only ASSOCIATEOID is presently supported. See "Alternate Worker ID's (alternateWokerIDs and alternateActorWokerIDs)" for property information.	No
alternateActorWokerIDs	Used to identify the Employee Position belonging to the Punch Edit being made (Supervisor Edit) that might belong to a Punch using an alternative element, suchas an ASSOCIATEOID. See "Alternate Worker ID's (alternateWokerIDs and alternateActorWokerIDs)" for property information.	No

Alternate Worker ID's (alternateWokerIDs and alternateActorWokerIDs)

Element Name	Description	Required?
idValue	Value for the Alternate Worker ID being supplied.	Yes
schemeCode.codeValue	Identifies the identifying element/item being supplied. At this time, only "ASSOCIATEOID" is supported.	Yes

Chapter 3

Data Collection Entries Web Service Method – Status - GET

The data-collection-entries.process/{eventid} HTTP GET method reports on the status of a previously submitted data-collection-entries.process HTTP POST punch uploaded payload method request.

Request

URI Format	/events/time/v1/data-collection-entries.process/{eventid} note eventID need to be fetch from POST API response body
METHOD	GET

HTTP Headers	**Authorization:** Bearer {accessToken}

Request Header Parameters

Header Name	Sample value	Comments
Authorization	Bearer {accessToken}	The Token Type (accessToken—i.e. "Bearer") plus the Access Token (access_token—i.e. "edc3c2-6b5d-4251-b258-d1208cbf6339") returned from ADP Marketplace's Token web service.

Possible Responses

201 - Succesful Import Status Response

This response indicates all the punch records in the payload completed, imported and posted successfully.

API:	/events/time/v1/data-collection-entries.process/{event-id}
Method:	Get
Condition:	This is get call with eventid which is part of confirm message for post call made earlier.
GitHub Sample Request/Sample Response Payload:	**Response:** success-response.json

207– Partial Failures

This response occurs when some punch records fail to import or post to the timecard.

API:	/events/time/v1/data-collection-entries.process/{event-id}
Method:	Get
Condition:	This is get call with eventid which is part of confirm message for post call made earlier.
GitHub Sample Request/Sample Response Payload:	**Response:** Get api response Partialfailure.json

400 – Transaction Failure Response

This response occurs when all the punches failed to import or post to the timecard.

API:	/events/time/v1/data-collection-entries.process/{event-id}
Method:	Get
Condition:	This is get call with eventid which is part of confirm message for post call made earlier.
GitHub Sample Request/Sample Response Payload:	**Response:** Get api failure response.txt

404 – Resource Not Found Response

This response occurs when the requested upload event is not found in the database.

API:	/events/time/v1/data-collection-entries.process/{event-id}
Method:	Get
Condition:	This is get call with eventid which is part of confirm message for post call made earlier.
GitHub Sample Request/Sample Response Payload:	**Response:** get api 404 response.txt

Chapter 4

Message Code Dictionary – userMessage.CodeValue

The example below includes a subset of possible error messages that be assigned to the "userMessage.codeValue" property that can be included in a response. The full list of error codes are defined in the table that follows this example response.

HTTP Headers	Content-Type: application/json;charset=UTF-8 Status Code: 400

Sample Response Body	"confirmMessage": {
	"createDateTime": "2016-11-10T17:00:43-05:00",
	"protocolStatusCode": {
	"codeValue": "400"
	},
	"protocolCode": {
	"codeValue": "http"
	},
	"requestStatusCode": {
	"codeValue": "failed"
	},
	"requestMethodCode": {
	"codeValue": "GET"
	},
	"processMessages": [{
	"processMessageID": {
	"idValue": "1"
	},
	"messageTypeCode": {
	"codeValue": "info"
	},

HTTP Headers	Content-Type: application/json;charset=UTF-8 Status Code: 400


```
"userMessage": {
"codeValue": "info.DCE_IMP_TOTALCOUNT",
"title": "Punch Import Statistics - Payload Total",
"messageTxt": "3"
}
}, {
"processMessageID": {
"idValue": "1"
},
"messageTypeCode": {
"codeValue": "info"
},
"userMessage": {
"codeValue": "info.DCE_IMP_FAILEDCOUNT",
"title": "Punch Import Statistics - Failed to Import",
"messageTxt": "3"
}
}, {
"processMessageID": {
"idValue": "1"
},
"messageTypeCode": {
"codeValue": "info"
},
"userMessage": {
"codeValue": "info.DCE_IMP_INPROCESSCOUNT",
"title": "Punch Import Statistics – In-Process",
"messageTxt": "0"
}
}, {
"processMessageID": {
"idValue": "1"
},
"messageTypeCode": {
"codeValue": "info"
},
"userMessage": {
"codeValue": "info.DCE_PST_TOTALCOUNT",
"title": "Punch Import Statistics - Total Processed",
"messageTxt": "0"
}
}, {
"processMessageID": {
"idValue": "1"
},
"messageTypeCode": {
"codeValue": "error"
},
"sourceLocationExpression": "events[0].data.transform.dataCollectionEntries[?(@.itemID='1')]].workerDataCollectionEntries[?(@.entryID='0')]",
"userMessage": {
"codeValue": "err.DCE_IMP_ACTIONCODEINVALID",
"title": "Punch Import Message",
"messageTxt": "Invalid ActionCode|actionCode=CLOCKFIN|badgeID=100182742|itemID=1|entryID=0"
}
}, {
"processMessageID": {
"idValue": "1"
},
"messageTypeCode": {
"codeValue": "error"
},
"sourceLocationExpression":
"events[0].data.transform.dataCollectionEntries[?(@.itemID='2'].workerDataCollectionEntries[?(@.entryID='0')]",
"userMessage": {
"codeValue": "err.DCE_IMP_ENTRYDATEMISSING",
"title": "Punch Import Message",
"messageTxt": "Invalid or missing entryDateTime|actionCode=TR4|badgeID=100182742|itemID=2|entryID=0"
}
}, {
"processMessageID": {
"idValue": "1"
},
"messageTypeCode": {
"codeValue": "error"
},
"sourceLocationExpression": "events[0].data.transform.dataCollectionEntries[?(@.itemID='1')]].workerDataCollectionEntries[?(@.entryID='0')]",
"userMessage": {
"codeValue": "err.DCE_IMP_BADGEMISSING",
"title": "Punch Import Message",
"messageTxt": "Missing badge value|itemID=3|entryID=0|actionCode=IN"
```

HTTP Headers	Content-Type: application/json;charset=UTF-8 Status Code: 400

	}
	}
	],
	"resourceMessages": [{
	"resourceMessageID": {
	"idValue": "f6a19fdc0b2445de882976f63f2672b8"
	},
	"resourceStatusCode": {
	"codeValue": "failed"
	},
	"resourceLink": {
	"rel": "alternate",
	"href": "/events/time/v1/data-collection-entries.process/d6254a05882a4461969f3c609c86e4ac",
	"method": "GET",
	"mediaType": "application/json",
	"encType": "UTF-8"
	}
	}
	]
	}
	}

Errors

Message ID	Description
Message ID	Description
exp_NoMobilePositionsFound	No position is associated with the Associate or Badge value provided.
err_BadgeReqForClockEmp	Badge value is missing and no AOID has been provided.
err_InvalidTimePunchCode	Specified Time Punch code is not recognized.
err_InvalidDateValue	Invalid date or unrecognized date format.
exp_PunchTimeOut	The punch is not within the six minute system time window. Provide a date/time within six minutes of the current date/time value. Applicable for systems with this option enabled.
exp_NoMobilePositionsFound	No position is associated with the provided ASSOCIATEOID.
err_DCE_IMP_BADGEMISSING	Missing badge value.
err_DCE_IMP_ACTIONCODEINVALID	Invalid action code.

Message ID	Description
err_DCE_IMP_ENTRYDATEMISSING	Invalid or missing entryDateTime.
err_DCE_IMP_DEVICEDATEMISSING	Invalid or missing deviceDateTime.
info_DCE_IMP_TOTALCOUNT	Bulk Punch Statistic Import Statistics (total record count found in the payload). Typically, only one message of this type occurs in the response.
info_DCE_IMP_FAILEDCOUNT	Bulk Punch Statistic Import Statistics (number of punches failing to import). Typically, only one message of this type occurs in the response.
info_DCE_IMP_FAILEDCOUNT	Bulk Punch Statistic Import Statistics (number of punches failing to import). Typically, only one message of this type occurs in the response.
info_DCE_IMP_INPROCESSCOUNT	Bulk Punch Statistic Import Statistics (number still in process). Typically, only one message of this type occurs in the response.
info_DCE_PST_TOTALCOUNT	Bulk Punch Statistic Posting Statistics (total record count considered for Posting). Typically, only one message of this type occurs in the response. This message may not exist at all if all punch records failed to import.
info_DCE_PST_FAILEDCOUNT	Bulk Punch Statistic Posting Statistics (number of punch records that failed to post to the timecard). Typically, only one message of this type occurs in the response. This message may not exist at all if all punch records imported into the system without any errors.
err_DCE_PST_COMMON	Indicates that the error is associated with the attempt to post the successfully imported time punch record to the timecard (i.e. while the import of the punch record succeeded the punch record failed to reach the employee's timecard).

Chapter 5

Frequently asked questions (FAQs):

Question 1: What if the clock isn't syncing when you are trying to do punches within 2 mins.Meaning as an example,clock in at 11:10 AM EST and clock out at 11:12 AM EST ?

POST	events/time/v1/data-collection-entries.process	Practitioner

Ans: If so, the punches may be following the 2+ minute rule. That rule gets applied when punches with timestamps within 3 minutes of each other may be considered duplicate punches or clocking mistakes and, as such, will not post to the timecard.

Steps to be followed if a partner wants to adjust the range down to 1 minute instead of 2.

Instructions:

- Navigate to: Setup>Time & Attendance>Additional Configuration>Company Preferences

- On the Company Preferences page from the "Application Preferences" side-bar tab there is an option named, "Time Punch Duplication Range Duration".
- You can try to adjust the range down to 1 minute instead of 2.
- The value selected is really a +1 value (meaning anything between 1 and 2 minutes might be considered a duplicate punch).
- The default is usually set to 2 (meaning, it's really a +2 value...anything between 2 - 3 minutes might be considered a duplicate punch).
- Note: that because of the seconds that are associated with the original punch, setting this to 1 may still not resolve all use cases for the clocking scenario described (all seconds are truncated when they are posted to the timecard but the seconds are used in the original duplication evaluation by the Posting Processor).