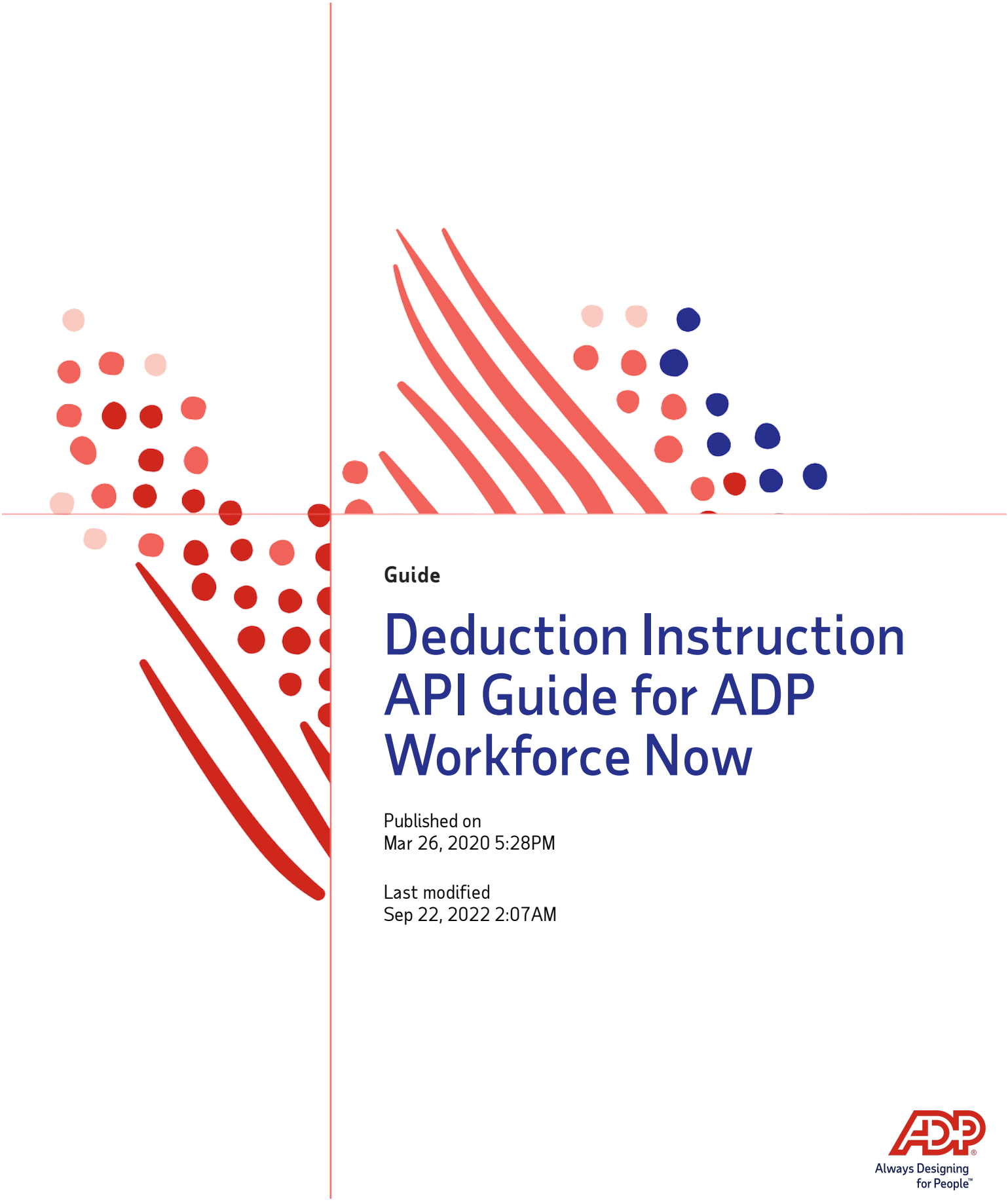


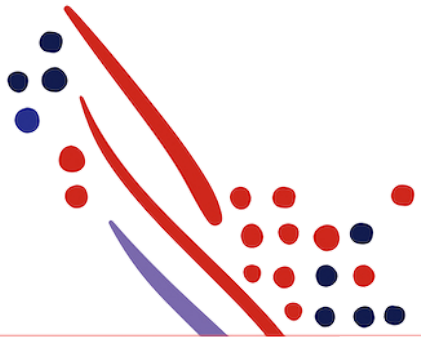
Guide

Deduction Instruction API Guide for ADP Workforce Now

Published on
Mar 26, 2020 5:28PM

Last modified
Sep 22, 2022 2:07AM

Always Designing for People™



ADP Copyright Information

Published on
Mar 26, 2020 5:28PM

Last modified
Sep 22, 2022 2:07AM

Table of Contents

- Impacted APIs
- Issue Description
- Suggested Work Around

US1168368: Retrieve additional data about deduction codes in meta

- Impacted APIs
- Issue Description
- Suggested Work Around

US1395538: The goal ACCRUED, and ADJUSTMENT fields are not supported from the API

- Impacted APIs
- Issue Description
- Suggested Work Around

US1395551: You cannot update a Deduction Status to Inactive or Active from the API

- Impacted APIs
- Issue Description
- Suggested Workaround

Chapter 1

About this Guide

Summary

The Deduction Instruction Application Programming Interfaces (APIs) are used to do the following:

- Start, stop, or change the payroll deduction setup for a worker.
- View all deduction instructions of a worker.
- Retrieve and send employee deductions to ADP (commonly used by Benefits Solutions).
- Capture taxable employer contributions.

Note

Deductions are now supported for Canadian and Cross Border clients. There is no change in request payload or the ADP Workforce Now user interface (UI).

Access tokens need to be used based on the following clients:

- United States (U.S.)
- Canadian
- Cross Border

What's New in this Guide?

In this section, we will announce any new revisions to the Deduction Instructions API and this guide.

September 2022

- Updated API endpoints and samples to link to API explorer.

March 2021

- Removed a limitation under chapter 8, "WFNWATERVI-49045: Worker Payroll Instruction GET call is missing the position status for the employee"

Feb 2021

- Added a limitation under chapter 8, "WFNWATERVI-49045: Worker Payroll Instruction GET call is missing the position status for the employee"

June 2020

- Added new items on **InactiveIndicator** support under the **Responses** section in both Chapter 3 - Use Case: Using the General Deduction Instruction Start API and Chapter 4: Changing a General Deduction Instruction.
- Added **Sample Event Notification Response** column with GitHub response links in the table within Chapter 6 - Use Case: Synching Worker Deductions Using Notifications.
- Removed a limitation in Chapter 8 - Known Issues and Limitations under **US1395551: You cannot update a Deduction Status to Inactive or Active from the API**.

About Deduction Categories

Currently, the ADP Workforce Now Deduction Instruction API only handles the Pre-tax and Post-tax deduction categories, which are defined as follows:

- **Pre-tax** – When you select the Pre-tax deduction category, select and complete an authorization form to request ADP to set up your Pre-tax Deduction Code. After you complete the form, ADP will set up your deduction and notify you when it's ready to use. Use the Pre-tax deduction category to set up Deduction Codes for the following:
    - Medical plans
    - Cafeteria 125 plans
    - 401(k) plans
    - Tax grids
    - Special compensation
    - Other similar deductions
- **Post-tax** – Used to set up Deduction Codes for the following:
    - Union Dues
    - 401(k)
    - Form W-2 (this displays by law)
    - Special Compensation
    - Benefit Plan
    - Other similar deductions

Deduction Codes could also be used to record and report on the Employer Contribution. For this type of deduction, the client practitioner will need to work with an ADP account manager or client service representative (depends on the service model) to create a special calculation that takes the employee deduction *-1.00, resulting in an earnings code. In other words, a negative deduction amount results in added earnings into the employee's paycheck, which is taxed and removed. Therefore, the employee does not actually receive the money.

Note

The Deduction Instruction APIs do not currently handle the following categories:

- **Direct Deposit** – Used to set up Deduction Codes for employee direct deposits. You must set up the Direct Deposit feature on the **Company Options** page or you cannot select this deduction category.
- **School District Tax** – Used to set up Deduction Codes for employees only in the state of Ohio. If a school district tax deduction was already set up for your company, this deduction category is not displayed (there is only one school district tax code per company).
- **Medicare Surtax** – Used to set up Deduction Codes for the Medicare Surtax for High Wage Earners, an employee tax that was approved by the United States Supreme Court on June 28, 2012. This tax is assessed at 0.9% of Medicare Subject Wages over $200,000 effective January 1, 2013. Medicare Surtax is reported separately on Form W-2 and Form 941. There is no employer tax component. If a Medicare Surtax deduction was already set up for your company, this deduction category is not displayed (there's only one Medicare Surtax Deduction Code per company). This code is never assigned to an employee. It's used when you adjust manual checks.
- **Liens** (Tax Levy, Garnishment, Support Order, Bankruptcy) – Used to set up Deduction Codes for employee liens. Displayed only for clients who use ADP's Wage Garnishment Processing Service (WGPS). If your client does not use WGPS, they can set up liens as an employee deduction or you can call your ADP service team to set up WGPS for your company.

Additional Information for Integration

You may want to retrieve actual deductions resulted after the payroll run. The following are options you and the client could use:

- **API based integration** – Your application could use the Payroll Data Output APIs to retrieve payroll output data and get the deduction information for each employee after each payroll job.
- **ADP Reporting** – If the client has an ADP Report, the client could create a report to pull in **PayDate**, **Deduction Code**, **Deduction Amount**, and **EE Name**. This is an existing process that people use to work with benefit vendors to check and balance payments to benefit vendors. Do the following:
    1. Log into the ADP Workforce Now user interface (UI) as a practitioner.
    2. Select **Reports > All Reports > Sample**.
    3. **Result:** There is a Benefit Deductions report where clients should be able to specify the Deduction Codes for the plans in which they are interested.
- **File base integration** – Ask the client to provide a payroll register.

Required Setup Steps

After your client acquires your data connector application, they need to make sure the Deduction Codes used by your application are set up in ADP Workforce Now.
When setting up Deduction Codes in ADP Workforce Now, the following also need to be configured. Afterwards, the configuration will apply to all workers.

- **Frequency** – Could be set to one of the following:
    - **OTO** - One Time Offer
    - **EPR** - Every Payroll or Process
- **Accumulators** – Used to accumulate the amounts for a Deduction Code. You can setup accumulators that are cleared several different times. For example, there is an accumulator for:
    - Year to Date
    - Quarter to Date
    - Several others, as needed

Tip

For Deduction Codes 81 through 96, the value is shown as a percentage. For Deduction Codes other than 81 through 96, the value is shown as an amount. Therefore, your application should be built similarly.

Chapter 2

Use Case: Retrieving Payroll Instructions (Payroll Instructions API)

Use Case Description

This use case returns the list of all available payroll instructions the requester is authorized to view for a given worker.

This API exposes values found in the ADP Workforce Now UI by selecting **People > Pay > Pay Profile > Deductions**. You must provide the Associate Organization ID (AOID) of the desired worker in the AOID Uniform Resource Identifier (URI) parameter.

Tip

To retrieve the ADP worker's unique Associate Object Identifier (AOID), your application can first retrieve all workers using GET /hr/v2/workers and creating the mapping between the two systems.

For more details, see the Worker Information Management API Guide for ADP Workforce Now.

API Usage

For API related information please refer Worker Payroll Instructions under API Explorer.

Method	URI	Description	Note
GET	/payroll/v1/workers/{aoid}/ payroll-instructions	Returns the list of all active payroll instructions for an employee until the current effective date.	ADP Workforce Now supports workers with multiple positions. When a worker has multiple positions, the Payroll Instructions API returns all active deductions and garnishments for all payroll file numbers for a given worker. Use the GET Workers API to retrieve employee position status.
GET	/payroll/v1/workers/{aoid}/payroll-instructions/{payroll-instruction-id}	Returns deductions based on the **pay-distribution-id**. **Tip:** When an associate has more than one position, each position has a unique **payrollGroupCode** and **payrollFileNumber**. The user can retrieve all the payroll instructions for all positions. If the user wants to retrieve payroll instructions for only one position, the user can make use of the GET /payroll/v1/workers/{aoid}/payroll-instructions/{payroll-instruction-id}. The API is called by passing the {payroll-instruction-id} of the particular position.	**Important:** Currently, this functionality is not working for Canada and cross border clients.
GET	/payroll/v1/workers/{aoid}/payroll-instructions ?asOfDate={YYYY-MM-DD}	Returns the list of all active deductions of an employee until the effective date provided in the **asOfDate** parameter.	
GET	/payroll/v1/workers/{aoid}/payroll-instructions ?Inactive=true	Returns the list of all active and inactive deductions for an employee until the current effective date.	**Important:** Currently, this functionality is not working for Canada and cross border clients.
GET	/payroll/v1/workers/{aoid}/payroll-instructions ?asofDate={YYYY-MM-DD}&Inactive=true	Returns the list of all active and inactive deductions for an employee until the effective date provided in the **asOfDate** parameter.	

Note

Parameter 'Inactive=true' returns both Inactive and active deductions. There is a similar behavior in the UI.

Application Scope

The canonical URI corresponding to the API needs to be added in the Consumer Application Registry (CAR) for the subscription following which a user can access the Deduction Instruction API and make successful API calls.

The following canonical needs to be added to your application scope to enable this use case:

/payroll/payrollManagement/payrollInstructionManagement/workerPayrollInstructionManagement/workerPayrollInstructions.read

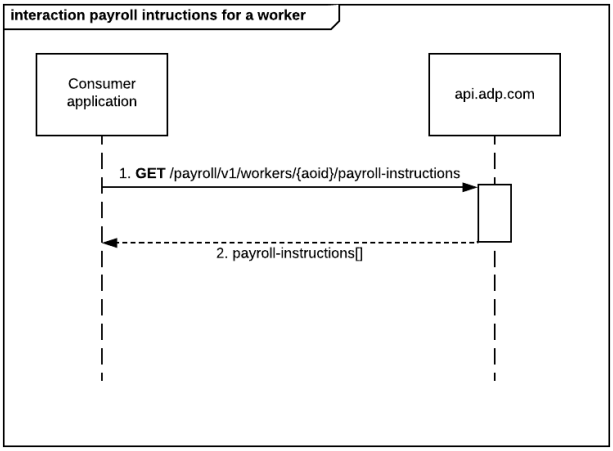
Supported Actors

Request Parameter roleCode Value	Usage
practitioner	Retrieves payroll instructions as a practitioner. A practitioner can view, start, stop, or change a deduction. A system user is considered a practitioner.

Request Header Parameters

None, in addition to the ADP standard header parameters.

Sequence of Interactions



The following are the steps shown in the previous diagram:

1. Your consumer application makes a request to the ADP API endpoint to GET payroll-instructions for a worker.
2. The ADP API endpoint responds to your consumer application with single worker payroll information.

Data Dictionary

The fields listed in the following table can be found in the ADP Workforce Now UI by selecting **People > Pay > Deductions**.

Payroll API Response Location	Field Name	Note
/payrollInstructions/generalDeductionInstructions	Not Displayed	
/payrollInstructions/generalDeductionInstructions/itemID	Not Displayed	Indicates the deduction Item ID. Each deduction has one item ID generated.
/payrollInstructions/generalDeductionInstructions/deductionCode/codeValue	**Deduction Code**	Indicates the Deduction Code.
/payrollInstructions/generalDeductionInstructions/deductionRate/rateValue	**Deduction Amount**	Indicates the deduction amount.
/payrollInstructions/generalDeductionInstructions/deductionStartDate	**Effective Date**	Indicates the deduction effective date.
/payrollInstructions/generalDeductionInstructions/deductionGoal/goalID	**Deduction Goal**	Indicates the Goal ID.
/payrollInstructions/generalDeductionInstructions/deductionGoal/goalBalanceAmount/amountValue	**Balance**	Indicates the Goal balance amount.
/payrollInstructions/generalDeductionInstructions/deductionGoal/goalLimitAmount/amountValue	**Deduction Goal Limit**	Indicates the Goal limit amount.
/payrollInstructions/generalDeductionInstructions/inactiveIndicator	Not Displayed	Indicates the status of the deduction. • **True** indicates active status. • **False** indicates Inactive status.
/payrollInstructions/garnishmentInstructions	Not Displayed	
/payrollInstructions/garnishmentInstruction	Not Displayed	Indicates the garnishment item ID. Each garnishment has one itemID generated.

s/itemID		
/payrollInstructions/garnishmentInstructions/workerLien	Not Displayed	
/payrollInstructions/garnishmentInstructions/workerLien /itemID	Not Displayed	Indicates the workerLien ID.
/payrollInstructions/garnishmentInstructions/workerLien /lienNumber	**Lien Number**	Indicates the worker Lien number.
/payrollInstructions/garnishmentInstructions/workerLien /lienTypeCode/codeValue	Not Displayed	Indicates the lien type code.
/payrollInstructions/garnishmentInstructions/workerLien /lienTypeCode/shortName	**Lien Type**	Indicates the lien name. For example, Garnishment.
/payrollInstructions/garnishmentInstructions/workerLien /lienSubTypeCode/codeValue	Not Displayed	Indicates the lien code value.
/payrollInstructions/garnishmentInstructions/workerLien /lienSubTypeCode/shortName	**Lien Subtype**	Indicates the lien subtype name.
/payrollInstructions/garnishmentInstructions/workerLien /lienStatusCode/effectiveDate	**Lien Start Date**	Indicates the lien start date.
/payrollInstructions/garnishmentInstructions/workerLien /lienStatusCode/codeValue	Not Displayed	
/payrollInstructions/garnishmentInstructions/workerLien /lienStatusCode/shortName	**Lien Status**	Indicates the lien status. Values are **Active**, **Complete**, and **Inactive**.
/payrollInstructions/garnishmentInstructions/workerLien /courtOrder/caseNumber	**Case # / Lien ID**	Indicates the Case or Lien ID.
/payrollInstructions/garnishmentInstructions/workerLien	Not Displayed	

/courtOrder/issuingJurisdiction		
/payrollInstructions/garnishmentInstructions/workerLien/courtOrder/issuingJurisdiction/countrySubdivisionLevel1	**Court Order State**	Indicates the court order state.
/payrollInstructions/garnishmentInstructions/workerLien/courtOrder/issuingJurisdiction/countrySubdivisionLevel1/subdivisionType	**City**, **State**, **ZIP**	Indicates the City, State, and ZIP.
/payrollInstructions/garnishmentInstructions/workerLien/courtOrder/issuingJurisdiction/countrySubdivisionLevel1/codeValue	Not Displayed	
/payrollInstructions/garnishmentInstructions/workerLien/courtOrder/issuingJurisdiction/countrySubdivisionLevel1/shortName	Not Displayed	
/payrollInstructions/garnishmentInstructions/workerLien/courtOrder/lienMaritalStatusCode	Not Displayed	
/payrollInstructions/garnishmentInstructions/workerLien/courtOrder/lienMaritalStatusCode/codeValue	Not Displayed	
/payrollInstructions/garnishmentInstructions/workerLien/courtOrder/lienMaritalStatusCode/shortName	**Marital Status**	Indicates the marital status.
/payrollInstructions/garnishmentInstructions/workerLien/fullServiceClientIndicator	Not Displayed	
/payrollInstructions/garnishmentInstructions/workerLien/allowanceQuantity	Not Displayed	
/payrollInstructions/garnishmentInstructions/workerLien	Not Displayed	

/fundsDisbursement		
/payrollInstructions/garnishmentInstructions/workerLien/fundsDisbursement/disbursementFees	**Funds Disbursement Schedule**	Indicates the disbursement fees.
/payrollInstructions/garnishmentInstructions/workerLien/fundsDisbursement/disbursementMessages	**Funds Disbursement Check Stub Messages**	
/payrollInstructions/garnishmentInstructions/garnishmentAmount		
/payrollInstructions/garnishmentInstructions/garnishmentAmount/amountValue	**Deduction Amount**	Indicates the garnishment amount.
/payrollInstructions/garnishmentInstructions/garnishmentStartDate	**Deduction Effective On**	Indicates the garnishment effective date.
/payrollInstructions/payrollFileNumber	Not Displayed	
/payrollInstructions/payrollAgreementID	Not Displayed	
/payrollInstructions/payrollGroupCode	Not Displayed	
/payrollInstructions/payrollGroupCode/codeValue	Not Displayed	
/payrollInstructions/payrollGroupCode/shortName	Not Displayed	
/payrollInstructions/workAssignmentStatus	Not Displayed	
/payrollInstructions/workAssignmentStatus/effectiveDateDate	**Effective Date**	
/payrollInstructions/workAssignmentStatus/codeValue	**Status**	

Responses

For API related information please refer Worker Payroll Instructions under API Explorer.

Response Code	Response Condition	message txt	Note
200 OK	Retrieves all the payroll deductions until the current effective date.		
200 OK	Retrieves all the payroll deductions until the provided **asOfDate** parameter.		
200 OK	Retrieves all the payroll deductions until the provided **asOfDate** parameter.		
200 OK	Retrieves any specific deduction passing deduction **itemID**.		
200 OK	Retrieves all the inactive deductions until the current **effectiveDate**.		
200 OK	Retrieves all the inactive deductions until the future **effectiveDate**.		
200 OK	Verifies the response when the employee does not have any deduction.		
200 OK	Verifies the response for future date when the employee does not have any deductions.		
204 No Content	Verifies the response when the employee does not get paid from ADP Workforce Now.		There will not be any response in this case.
400 Bad Request	Verifies the response when the **itemID** passed in the URI is invalid.		
500 Internal Server Error	Verifies the response when the associate ID passed in the request URI is invalid.	"userMessage": {"messageTxt": "Label not found, missing following key: Exception in the requestHTTP 404 Not Found"}	

Tip

To update deductions for a given worker AOID using the POST API (as described in Chapter 3 - Use Case: Using the General Deduction Instruction Start API, Chapter 4 - Use Case: Changing a General Deduction Instruction, and Chapter 5 - Use Case: Using the General Deduction Instruction Stop API the payload needs to send a few values. These values can be obtained only by the GET /payroll/v1/workers/{associateoid}/payroll-instructions API. The following values need to be used from the response of the Payroll Instructions API:

- payrollFileNumber
- payrollAgreementID
- itemID

Use Case: Using the General Deduction Instruction Start API

Use Case Description

This use case is used to start a general deduction instruction for a worker.

API Usage

For API related information please refer Worker Payroll Instructions under API Explorer.

Method	URI	Description
POST	/events/payroll/v2/worker-general-deduction-instruction.start	Starts worker general deduction instruction information.
GET	/events/payroll/v2/worker-general-deduction-instruction.start/meta	Returns an event metadata. **Note:** Sample payload can be built from the meta call.

Important

- In meta response for **DeductionGoal > Goal ID**, the codeList values are not available. For now, it's hardcoded from (1 to 9).
- **sub-resource**, with the value of "null", is optional and not in use.
- **sub-resource**, **payrollInstructionGeneralDeductionInstructionDeductionGoalGoalStartDate**, is optional and not in use.

In the given payload (deduction-start-currentdate-200.request.json), the following is the purpose of the eventContext and transform sections:

- **eventContext:** A set of keys, identifying the subject. In the payload, the **associateID** field is present under **eventContext**. The **associateID** identifies the subject.
- **transform:** Provides the values added or changed with respect to the subject keys defined in the **eventContext** section.

Application Scope

The following canonical needs to be added to your application scope to enable this use case:

/payroll/payrollManagement/payrollInstructionManagement/workerGeneralDeductionInstructionManagement/workerGeneralDeductionInstruction.start

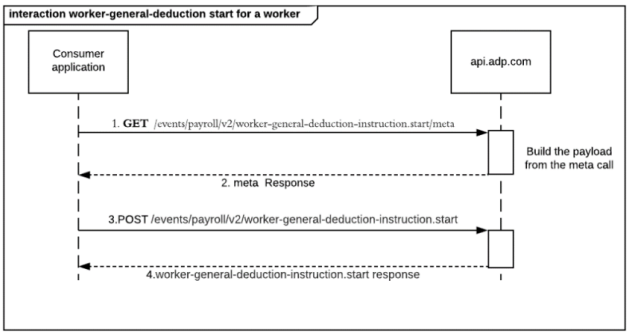
Supported Actors

Request Parameter roleCode Value	Usage
practitioner	Starts deduction instructions for workers.

Request Header Parameters

None, in addition to the ADP standard header parameters.

Sequence of Interactions



The following are the steps shown in the previous diagram:

1. Your consumer application makes a request to the ADP API endpoint for **Worker-general-deduction-instruction** start meta.
2. The ADP API endpoint responds to your consumer application with the meta payload, which lists eligible deductions for the worker under the **/meta/data/transforms/ payrollInstruction/generalDeductionInstruction/deductionCode/codeList** collection. Your consumer application needs to validate that the Deduction Code processed is returned in the codeList collection before building the payload for the next call. If the Deduction Code you need to start is not part of the collection, your application needs to provide user error handling and stop or go to the next step.
3. Your consumer application makes a request to the ADP API endpoint for the **Worker-general-deduction-instruction** start API with the payload.
4. The ADP API endpoint responds to the consumer application about the details of the **Worker-general-deduction-instruction** start.

Data Dictionary

The fields listed in the following table can be found in the ADP Workforce Now UI by selecting People > Pay > Deductions.

Schema Location	ADP Workforce Now Field Name	Required (Y/N)?	Note
/events/data/eventContext	Not Displayed	N	
/events/data/eventContext/worker	Not Displayed	N	

/events/data/eventContext/worker /associateOID	Not Displayed	Y	
/events/data/eventContext/payrollInstruction	Not Displayed	N	
/events/data/eventContext/payrollInstruction /payrollGroupCode	Not Displayed	Y	The **payrollGroupCode** found in payroll. This is an Instruction GET call.
/events/data/eventContext/payrollInstruction /payrollGroupCode /codeValue	Not Displayed	Y	
/events/data/eventContext/payrollInstruction /payrollGroupCode /shortName	Not Displayed	Y	
/events/data/eventContext /payrollInstruction/payrollFileNumber	Not Displayed	Y	The **payrollFileNumber** found in payroll. This is an instruction GET call.
/events/data/eventContext/payrollInstruction /payrollAgreementID	Not Displayed	Y	The **payrollAgreementID** found in payroll. This is an instruction GET call.
/events/data/effectiveDateTime	**Effective Date**	Y	
/events/data/payrollInstruction	Not Displayed	N	
/events/data/payrollInstruction /generalDeductionInstruction	Not Displayed	N	
/events/data/payrollInstruction /generalDeductionInstruction/deductionCode	Not Displayed	Y	
/events/data/payrollInstruction/generalDeductionInstruction/deductionCode /codeValue	Not Displayed	Y	
/events/data/payrollInstruction /generalDeductionInstruction/deductionCode /shortName	**Deduction Code**	Y	
/events/data/payrollInstruction /generalDeductionInstruction/deductionRate	Not Displayed	N	

/events/data/payrollInstruction /generalDeductionInstruction/deductionRate /rateValue	**Deduction Amount**	Y	
/events/data/payrollInstruction /generalDeductionInstruction/deductionGoal	**Deduction Goal**	Y	
/events/data/payrollInstruction /generalDeductionInstruction/deductionGoal /goalID	Not Displayed	Y	
/events/data/payrollInstruction /generalDeductionInstruction/deductionGoal /goalLimitAmount	Not Displayed	N	
/events/data/payrollInstruction /generalDeductionInstruction/deductionGoal /goalLimitAmount/amountValue	**Deduction Goal Limit**	Y	
/events/data/payrollInstruction /generalDeductionInstruction/deductionGoal /goalLimitAmount/currencyCode	Not Displayed	Y	
/events/data/payrollInstruction /generalDeductionInstruction/deductionGoal /goalBalanceAmount	**Balance**	N	
/events/data/payrollInstruction /generalDeductionInstruction/deductionGoal /goalBalanceAmount/amountValue	Not Displayed	Y	
/events/data/payrollInstruction /generalDeductionInstruction/deductionGoal /goalBalanceAmount/currencyCode	Not Displayed	Y	

Responses

You may encounter exceptions outside your common success scenarios. You must account for these exceptions during your initial development.

For more information, see API Common Exceptions and Tips for Handling.

For API related information please refer Worker Payroll Instructions under API Explorer.

Response Code	Response Condition	message Txt	Tips to Handle
200 OK	When adding a deduction with **InactiveIndicator** as true.		
200 OK	When adding a deduction with **InactiveIndicator** as false.		
200 OK	**When adding a deduction without a deduction goal and the effective date as the current date.**	NA	
200 OK	**When adding a deduction with a deduction goal and the effective date as the current date.**	NA	
200 OK	**When adding a deduction with a deduction goal and effective date as the future date.**	NA	
400 Bad Request	Changing a deduction with InactiveIndicator as true and amount is not zero		
400 Bad Request	When adding a deduction keeping **InactiveIndicator=true**, and amount as some value Ex: 10		
400 Bad Request	**When trying to start a deduction but the goal limit amount is invalid.**		
400 Bad Request	**When trying to start a deduction but the deduction goal code is invalid.**		
400 Bad Request	**When trying to start a deduction but the deduction goal balance amount is invalid.**		
400 Bad Request	**When trying to start the deduction for the associate who already has this deduction.** **For example, Deduction Code: H.**	{"userMessage": {"messageTxt": "You cannot add this employee deduction because the deduction currently exists with the effective date of 08/20/2018." }}	**The specified deduction is already started. Make sure to check all the deductions are started for an associate. Try with a different deduction, which is not yet started.**
400 Bad Request	**When the invalid deduction code is passed in the request.**	{"userMessage": {"messageTxt": "deductionCodeis invalid."}}	**Make sure a valid supported Deduction Code is passed in the request.**
400 Bad Request	**When there is an invalid payrollGroupCode/codeValue in the request.**	{ "userMessage": {"messageTxt": "payrollGroupCode/codeValue is invalid." }}	**Make sure a valid payrollGroupCode /codeValue is passed in the request.**
400 Bad Request	**When there is an invalid payrollAgreementID in the request.**	{"userMessage": {"messageTxt": "payrollAgreementIDis invalid." }}	**Make sure a valid payrollAgreementID is passed in the request.**
400 Bad Request	**When there is an invalid payrollFileNumber in the request.**	{"userMessage": {"messageTxt": "payrollFileNumber is invalid." }}	**Make sure a valid payrollFileNumber is passed in the request.**
400 Bad Request	When a **DeductionAmount** is passed as	{"userMessage": {"messageTxt":	Make sure you have entered the

	an invalid value in the request.	"Deduction Amountfield must contain a non-zerovalue between -99,999,999.99and 99,999,999.99 with a decimalprecision of 2 places." }}	amount value correctly. **Note:** Deduction Amount must contain a non-zero value between-99,999,999.99 and 99,999,999.99 with a precision of 2.

Chapter 4

Use Case: Changing a General Deduction Instruction

Use Case Description

This use case changes a general deduction instruction for a single worker.

API Usage

For API related information please refer Worker Payroll Instructions under API Explorer.

Method	URI	Description
POST	/events/payroll/v2/worker-general-deduction-instruction.change	Changes a worker's general deduction instruction information.
GET	/events/payroll/v2/worker-general-deduction-instruction.change/meta	Returns event metadata. **Note:** Sample payload can be built from the meta call.

Application Scope

The following canonical needs to be added to your application scope to enable this use case.

API	Canonical URI
Change deductions	**/payroll/payrollManagement/payrollInstructionManagement/workerGeneralDeductionInstructionManagement/ workerGeneralDeductionInstruction.change**

Supported Actors

Request Parameter roleCode Value	Usage
Practitioner	Retrieves payroll instructions. A practitioner can view, start, stop, or change a deduction. A system user is considered a practitioner.

Request Header Parameters

Parameter Name	Required (Y/N)	Usage	Value
Content-Type	Y	application/json	**Content-Type:application/json**

Supported OData Query Options

There are no supported OData Query Options.

Other Supported Parameters

There are no other supported parameters.

Sequence of Interactions

The following are the steps shown in the previous diagram:

1. Your consumer application makes a request to the ADP API endpoint for the Worker-general-deduction-instruction change API.
2. The ADP endpoint responds to your consumer application with the details of the Worker-general-deduction-instruction change.

Data Dictionary

The fields listed in the following table can be found in the ADP Vantage HCM UI by selecting People > Pay & Taxes > Deductions.

Schema Location	Field Name in ADP Vantage HCM	Is Required (Y/N)	Note
/data/eventContext/worker/associateOID	Not Displayed	Y	The AOID can be retrieved from GET workers API call.
/data/eventContext/payrollInstruction/generalDeductionInstruction/itemID	**CODE**	Y	The **ItemID** can be retrieved from codelist API call.
/data/transforms/effectiveDateTime	**Effective Date or START DATE**	Y	
/data/transforms/payrollInstruction/generalDeductionInstruction/deductionRate/rateValue	**AMOUNT / PERCENT**	Y	
/data/transforms/payrollInstruction/	Not Displayed		

generalDeductionInstruction/deductionRate/unitCode			
/data/transforms/payrollInstruction/generalDeductionInstruction/deductionRate/currencyCode	Not Displayed		
/data/transforms/payrollInstruction/generalDeductionInstruction/deductionRate/unitCode	Not Displayed		
/data/transforms/payrollInstruction/generalDeductionInstruction/deductionRate/unitCode/ codeValue	Not Displayed		
/data/transforms/payrollInstruction/generalDeductionInstruction/deductionGoal/goalID	**Goal #**	N	Indicates the goal number. Valid values are 1-9.
/data/transforms/payrollInstruction/generalDeductionInstruction/deductionGoal/goalLimitAmount/amountValue	**GOAL AMOUNT**	Y	
/data/transforms/payrollInstruction/generalDeductionInstruction/deductionGoal/goalLimitAmount/currencyCode	Not Displayed	N	

Responses

You may encounter exceptions outside your common success scenarios. You must account for these exceptions during your initial development.

For more information, see API Common Exceptions and Tips for Handling.

For API related information please refer Worker Payroll Instructions under API Explorer.

Response Code	Response Condition	messageTxt	Tips to Handle
200 OK	Changing a deduction with InactiveIndicator as true		
200 OK	Changing a deduction with InactiveIndicator as false		

200 OK	When changing a deduction, without a deduction goal, with effective date as current.	NA	
200 OK	When changing a deduction with deduction goal with effective date as current date.	NA	
200 OK	When changing a deduction with the deduction goal effective on a future date.	N/A	
400 Bad Request	When trying to change a deduction but the goal limit amount is invalid.		
400 Bad Request	Trying to change a deduction but the deduction goal code is invalid.		
400 Bad Request	Trying to change a deduction but the deduction goal balance amount is invalid.		
400 Bad Request	When the invalid payroll **GroupCode/codeValue** is passed in the request.	{"userMessage": {"messageTxt": "payrollGroupCode/codeValue is invalid."}}	Make sure a valid **payrollGroupCode/codeValue** is passed in the request.
400 Bad Request	When the invalid **payrollFileNumber** is passed in the request.	{"userMessage": {"messageTxt": "payrollFileNumber is invalid."}}	Make sure a valid **payrollFileNumber** is passed in the request.
400 Bad Request	When the invalid **payrollAgreementID** is passed in the request.	{"userMessage": {"messageTxt": "payrollAgreementID is invalid."}}	Make sure a valid **payrollAgreementID** is passed in the request.
400 Bad Request	When the invalid itemId is passedin the request.	{"userMessage": {"messageTxt":"itemId is invalid."}}	Make sure a valid itemID is passed in the request. **Note:** The **itemID** is obtained from the Payroll Instructions API.
400 Bad Request	When an invalid **deductionCode** is passed in the request.	{"userMessage": {"messageTxt": "deduction Code is invalid."}}	Make sure a valid **deductionCode** is passed in the request.
400 Bad Request	When the future effective date is passed in the request without a changing deduction amount or deduction goal.	"userMessage": {"messageTxt": "To save, please change the Deduction Amount field and/or the Active field."}	

Chapter 5

Use Case: Using the General Deduction Instruction Stop API

Use Case Description

This use case is used to stop a general deduction instruction for a single worker. A successful stop request changes the Deduction record from the **Active** to the **Inactive** state.

To view **Inactive** deductions in the ADP Workforce Now UI, select **People > Pay > Pay Profile** and follow these steps:

1. Find the worker record.
2. Click **Deductions**.
3. Select data from **Show as of**.

4. Switch **SHOW INACTIVE** to **YES**.

Your program could also retrieve **Inactive** records using APIs. For more information, see Chapter 2 - Use Case: Retrieving Payroll Instructions (Payroll Instructions API).

API Usage

For API related information please refer Worker Payroll Instructions under API Explorer.

Method	URI	Description
GET	/events/payroll/v2/worker-general-deduction-instruction.stop/meta	Returns an event metadata. **Note:** Sample payload can be built from the meta call.
POST	/events/payroll/v2/worker-general-deduction-instruction.stop	Stops worker general deduction instruction information.

In the given payload (deduction-change-200.request.json), the following is the purpose of the **eventContex** and **transform** sections:

- **eventContext**: A set of keys, identifying the subject. In the payload, the **associateID** field is present under **eventContext**. The **associateID** identifies the subject.
- **transform**: Provides the values added or changed with respect to the subject keys defined in the **eventContext** section.

Application Scope

The following canonical needs to be added to your application scope to enable this use case:

/payroll/payrollManagement/payrollInstructionManagement/ workerGeneralDeductionInstructionManagement/ workerGeneralDeductionInstruction.stop

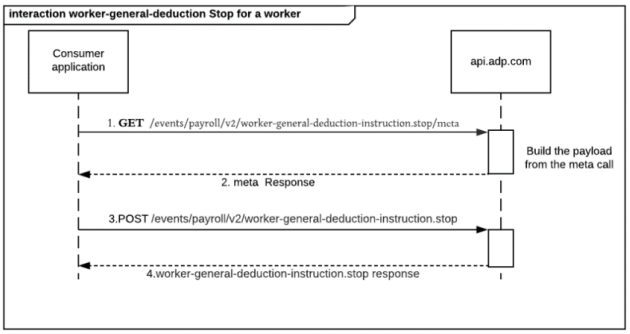
Supported Actors

Request Parameter roleCode Value	Usage
practitioner	Stops deduction instructions for workers.

Request Header Parameters

None, in addition to the ADP standard header parameters.

Sequence of Interactions



The following are the steps shown in the previous diagram:

1. Your consumer application makes a request to the ADP API endpoint for a **Worker-general-deduction-instruction Stop** meta.
2. The ADP API endpoint responds to your consumer application with the meta payload, which lists the eligible deductions for the worker under the **/meta/data/transforms/payrollInstruction/generalDeductionInstruction/deductionCode/codeList** collection. Your consumer application needs to validate the Deduction Code to be processed is returned in the **codeList** collection before building the payload for the next call. If the Deduction Code you need to start is not part of the collection, your application needs to provide user error handling and stop or go to the next step.
3. Your consumer application makes a request to the ADP API endpoint for the **Worker-general-deduction-instruction Stop** API with the payload.
4. The ADP API endpoint responds to the consumer application about the details of the **Worker-general-deduction-instruction Stop**.

Data Dictionary

The fields listed in the following table can be found in the ADP Workforce Now UI by selecting People > Pay > Deductions.

Schema Location	ADP Workforce Now Field Name	Required (Y/N)?	Note
/events/data/eventContext	Not Displayed	N	
/events/data/eventContext/worker	Not Displayed	N	
/events/data/eventContext/worker/associateOID	Not Displayed	Y	The **associateOID** will get this from the Worker Call API.
/events/data/eventContext/payrollInstruction	Not Displayed	N	
/events/data/eventContext/payrollInstruction /itemID	Not Displayed	Y	The **itemID** will get this from the payroll Instructions GET call.
/events/data/eventContext/payrollInstruction /payrollGroupCode	Not Displayed	Y	The **payrollGroupCode** will get this in the payroll Instructions GET call.
/events/data/eventContext/payrollInstruction /payrollGroupCode	Not Displayed	Y	

/codeValue			
/events/data/eventContext/payrollInstruction /payrollGroupCode /shortName	Not Displayed	Y	
/events/data/eventContext/payrollInstruction /payrollFileNumber	Not Displayed	Y	The **payrollFileNumber** will get this in the payroll Instructions GET call.
/events/data/eventContext/payrollInstruction /payrollAgreementID	Not Displayed	Y	The **payrollAgreementID** will get this in the payroll Instructions GET call.
/events/data/eventContext/payrollInstruction /generalDeductionInstruction	Not Displayed	Y	
/events/data/eventContext/payrollInstruction /generalDeductionInstruction/deductionCode	Not Displayed	Y	
/events/data/eventContext/payrollInstruction /generalDeductionInstruction/deductionCode /codeValue	Not Displayed	Y	
/events/data/eventContext/payrollInstruction /generalDeductionInstruction/deductionCode /shortName	Not Displayed	Y	
/events/data/transform/effectiveDateTime	**Effective Date**	Y	

Responses

You may encounter exceptions outside your common success scenarios. You must account for these exceptions during your initial development.

For more information, see API Common Exceptions and Tips for Handling.

For API related information please refer Worker Payroll Instructions under API Explorer.

Response Code	Response Condition	messageTxt	Tips to Handle
200 OK	When the request is processed successfully.	NA	
400 Bad Request	When the invalid **payrollFileNumber** is passed in the request.	{"userMessage": {"messageTxt": "payrollFileNumberis invalid."}}	Make sure a valid **payrollFileNumber** is passed in the request.
400 Bad Request	When the	{"userMessage": {"messageTxt":	Make sure a

	invalid **payrollAgreementID** is passed in the request.	"payrollAgreementIDis invalid."}}	valid **payrollAgreementID** is passed in the request.
400 Bad Request	When the invalid **itemID** is passed in the request.	{"userMessage": {"messageTxt": "itemIdis invalid."}}	Make sure a valid **itemId** is passed in the request.
400 Bad Request	When the **effectiveDate** is passed as previous year.	{"userMessage": {"messageTxt": "Effective date must besame as or laterthan position start date." }}	Make sure the **effectiveDate** is the same as or later than the position start date.

Chapter 6

Use Case: Synching Worker Deductions Using Notifications

ADP applications trigger notifications when data is updated, and your application could subscribe interested notifications to sync up data.

For more information, see the ADP Event API and Event Notification Guide for more information about ADP Notifications.

For API related information please refer Worker Payroll Instructions under API Explorer.

API	Scopes	Scopes to Subscribe
/events/payroll/v2/worker-general-deduction-instruction.start	**/payroll/payrollManagement/payrollInstructionManagement/workerGeneralDeductionInstructionManagement/workerGeneralDeductionInstruction.start**	**/payroll/payrollManagement/payrollInstructionManagement/workerGeneralDeductionInstructionManagement/workerGeneralDeductionInstruction.start.eventNotify.subscribe**
/events/payroll/v2/worker-general-deduction-instruction.change	**/payroll/payrollManagement/payrollInstructionManagement/workerGeneralDeductionInstructionManagement/workerGeneralDeductionInstruction.change**	**/payroll/payrollManagement/payrollInstructionManagement/workerGeneralDeductionInstructionManagement/workerGeneralDeductionInstruction.change.eventNotify.subscribe**
/events/payroll/v2/worker-general-deduction-instruction.stop	**/payroll/payrollManagement/payrollInstructionManagement/workerGeneralDeductionInstructionManagement/workerGeneralDeductionInstruction.stop**	**/payroll/payrollManagement/payrollInstructionManagement/workerGeneralDeductionInstructionManagement/workerGeneralDeductionInstruction.stop.eventNotify.subscribe**

Chapter 7

Frequently Asked Questions

Question 1: How does Goal Amount work for an employee?

Answer: If the goal amount is added to a deduction (for example, **D - DENTAL**), then the deduction **DENTAL** will be applied in payroll until the deduction amount reaches the goal limit amount. After it reaches that amount, the deduction will not be applied from the next payroll.

GOAL ID
There are total of nine goals (1 to 9) for every employee. Each goal is restricted to only one deduction. For example, if the goal ID: 1 is applied in deduction **DENTAL** than the same **Goal ID** cannot be applied to any other deduction.

See Chapter 3 - Use Case: Using the General Deduction Instruction Start API for more information.

Question 2: What is the difference between goalLimitAmount, goalBalanceAmount, and goalAccruedAmount?

Answer:

- **goalLimitAmount** – The maximum limit of a deduction. For example, if the deduction amount is $10 and the GoalLimitAmount is $100, then the deduction is applied in payroll until a total of $100 is deducted.
- **goalBalanceAmount** – The remaining amount from the goalLimitAmount until the current payroll.
- **goalBalanceAmount** – The deduction amount from the goalLimitAmount until the current payroll.

Question 3: If a deduction is already applied to an employee, how do you adjust the Overall Deduction Goal Limit?

Answer: In this case, user needs to include **GoalAccruedAmount**. If the deduction is already running during every payroll but now the user wants to add a goal amount, then the user should include Additional Amount Already Taken if it's from the UI or **goalBalanceAmount** if it is from API. This means the amount entered in this field has already been deducted.
After this is completed, the New Balance amount in the UI will be displayed, which will be auto-calculated as the new **goalLimitAmount**.

For example, if Deduction **DENTAL** is already running in every payroll and a total of $50 is deducted, the total **goalLimitAmount** is $500.

If the user wants to adjust the deduction goal limit amount, then $50 is entered in **Additional Amount Already Taken**.

After this is completed, the **New Goal Balance** is $450, which is automatically calculated.

Question 4: When is a deduction processed for an employee?

Answer:

- Case 1: If the deduction is added in current date, it will be effective from next payroll. Not from the current payroll.
- Case 2: If the deduction is added in future date, it will be effective from the next payroll from the date when it is active.

Question 5: How do I keep deduction records in synch between ADP Workforce Now and my (partner) application?

Answer:

1. To retrieve all deductions, including both future effective deductions, for both the **Active** and **Inactive** status, do the following (in order):
    - Retrieve all deductions for each employee using **asOfDate={effectiveDate} & inactive=true** for the enrollment. For more information, see Chapter 2 - Use Case: Retrieving Payroll Instructions (Payroll Instructions API).
    - For each deduction record in your (partner) system, compare the records retrieved from ADP Workforce Now. For example, the **inactiveIndicator**, **deductionStartDate**, and so on.
2. To START or CHANGE enrollment, do the following:
    - **Case 1: What if there are no effective deductions available?** – Add the deduction by using POST .start with the effective date and amount. For more information, see Chapter 3 - Use Case: Using the General Deduction Instruction Start API.
    - **Case 2: What if there are deductions available but they're Inactive?** - The API doesn't handle updating the deduction status to **Active** or **Inactive**. It can only be completed on the ADP Workforce Now user interface (UI).
    - **Case 3: What if deductions are available, are they the same deduction amount?** – Do nothing.
    - **Case 4: What if deductions are available and there are different deduction amounts?** – Update the deduction by using POST .change with the effective date and amount. For more information, see Chapter 4 - Use Case: Changing a General Deduction Instruction.
3. To Stop enrollment, do the following:
    - **Case 1: What if there are no effective deductions available?** Do nothing.
    - **Case 2: What if deductions are available but they're Inactive?** Do nothing.

- **Case 3: What if the deductions are available but they're Active?** Stop the deduction by using POST .stop with the effective date. For more information, see Chapter 4 - Use Case: Changing a General Deduction Instruction.

Question 6: How do I set up a deduction for an employer contribution?

Answer:

1. ADP clients will need to contact their ADP representative to create the deduction code. During the deduction set up at the company level, the client can specify an employer matching amount or percentage for the specific deduction code. When deductions start for an employee the employer contribution will automatically be handled based on the deduction code configuration.
2. If the employer contribution is not recurring, then the client has the option to set it up as a deduction memo.

Question 7: How do I make a read-only Deduction Code editable, so my application can use APIs to perform updates?

Answer:
Sometimes, a deduction is listed as read only. To view a deduction in ADP Workforce Now, select People > Pay Profile > Deductions (this page is read-only because the deduction is managed by the ADP benefit enrollment). To make a deduction editable there are two options:

- Setting could be updated at the Plan level by ADP Service Center Representatives for all employees – Under the Universal Access mode, select **Setup > Benefits > Plan Setup**. Find the plan to be released, select **Actions**, and click on **Suspend Payroll Deductions**, and click **OK** to confirm the change. After it's done, the deduction could then be updated through the API and UI for employees.
- Setting could also be updated by a client practitioner user for each impacted employee by doing the following:
    1. Go to **Setup > Benefits > Plan Setup**. Find the plan to be released. Under the **Eligible Group** column, expand the eligible group and click **Eligible Employees**.
    2. Navigate to the **Deductions** tab by selecting effective date.
    3. Choose the **Company Code** for which the changes need to be done. Deselect the deduction code, enter it as blank, and click **Next**.
    4. Click **Review** and validate the changes done.
    5. At the top of the page, click **Done**.

After it's done, the deduction can be updated through the API and UI for employees.

Chapter 8

Known Issues and Limitations

US1395531: In Payroll Instruction Response, effectiveDate field is displaying as effectiveDateDate

Impacted APIs

Method	URI	roleCode Value
GET	/payroll/v1/workers/{aoid}/payroll-instructions	practitioner

Issue Description

Within the GET API response payroll instruction, under **workAssignmentStatus**, the **effectiveDateDate** field should be displayed as **effectiveDate**.

Suggested Workaround

There is no workaround available.

US1390613: In Deduction Start/Change Request Payload, goalBalanceAmount is currently used for additional amount already taken in the ADP Workforce Now UI

Impacted APIs

Method	URI	roleCode Value
GET	/events/payroll/v2/worker-general-deduction-instruction.start/meta	practitioner
POST	/events/payroll/v2/worker-general-deduction-instruction.start	practitioner
GET	/events/payroll/v2/worker-general-deduction-instruction.change/meta	practitioner
POST	/events/payroll/v2/worker-general-deduction-instruction.change	Practitioner

Issue Description

In the request payload for deduction start and change, the following section is used for Additional Amount Already Taken in the ADP Workforce Now UI:

"deductionGoal": {"goalBalanceAmount": {"amountValue": 300}}

But, per the meta response, there is a field called **GoalAccruedAmount** to handle the **Additional Amount Already Taken** and **goalBalanceAmount** fields to handle the **Balance** field in the ADP Workforce Now UI.

So, to handle the **Additional Amount Already Taken** field in the ADP Workforce Now UI, the section should be as follows:

"deductionGoal": {"GoalAccruedAmount": {"AmountValue": 300}}

Suggested Workaround

There is no workaround available.

US1294444: Unable to retrieve the newly added Deduction Code with Deduction Frequency as OTO in the General Deduction Start/Change/Stop API meta response codelist

Impacted APIs

Method	URI	roleCode Value
GET	/events/payroll/v2/worker-general-deduction-instruction.start/meta	practitioner
POST	/events/payroll/v2/worker-general-deduction-instruction.start	practitioner
GET	/events/payroll/v2/worker-general-deduction-instruction.change/meta	practitioner
POST	/events/payroll/v2/worker-general-deduction-instruction.change	practitioner
GET	/events/payroll/v2/worker-general-deduction-instruction.stop/meta	practitioner
POST	/events/payroll/v2/worker-general-deduction-instruction.stop	practitioner

Issue Description

When a new deduction code of Post Tax Type is added with the deduction frequency of **OTO** in ADP Workforce Now, the newly added deduction code will not return in the General deduction start/change/stop API meta response codelist.

Suggested Work Around

The OTO effect could be achieved by starting a deduction then stopping the deduction after its being processed. A success processed deduction response contains the effective week and the Pay Date. Your application could use this data to stop the deduction.

US1168368: Retrieve additional data about deduction codes in meta

Impacted APIs

Method	URI	roleCode Value
GET	/events/payroll/v2/worker-general-deduction-instruction.start/meta	practitioner

GET	/events/payroll/v2/worker-general-deduction-instruction.change/meta	practitioner
GET	/events/payroll/v2/worker-general-deduction-instruction.stop/meta	practitioner

Issue Description

The following fields are not part of the **codeList** response for each deduction code as the value displayed in ADP Workforce Now within the table under **Validation Tables > Payroll > Deductions and Deposits > Deductions**:

- **Description**
- **Category**
- **Company Code**
- **Deduction Frequency**
- **Deduction Group**
- **Accumulator (Yes vs. No)**
- **Arrears (Yes vs. No)**
- **Status (Inactive/Active)**

Suggested Work Around

There is no workaround available.

US1395538: The goal ACCRUED, and ADJUSTMENT fields are not supported from the API

Impacted APIs

Method	URI	roleCode Value
GET	/events/payroll/v2/worker-general-deduction-instruction.start/meta	practitioner
POST	/events/payroll/v2/worker-general-deduction-instruction.start	practitioner
GET	/events/payroll/v2/worker-general-deduction-instruction.change/meta	practitioner
POST	/events/payroll/v2/worker-general-deduction-instruction.change	practitioner

Issue Description

Viewing and managing **GOAL ACCRUED**, and **ADJUSTMENT** are not supported from the API like an administrator could do in the ADP Workforce Now UI.

Suggested Work Around

There is no workaround available.

US1395551: You cannot update a Deduction Status to Inactive or Active from the API

Impacted APIs

Method	URI	roleCode Value
GET	/events/payroll/v2/worker-general-deduction-instruction.change/meta	practitioner
POST	/events/payroll/v2/worker-general-deduction-instruction.change	practitioner

Issue Description

The Deduction Instructions API cannot make a deduction status **Active** or **Inactive**. Only retrieve is possible through the payroll instruction GET API.

Suggested Workaround

You can make the deduction status **Active** or **Inactive** using the ADP Workforce Now UI.