TestSpace

1. About
2. Untitled
3. Test Adobe cursor jumping on tables
4. MedScout issue with boling and colors
5. Cadent TOC issue
6. 5.4.1 DocumentSearchRequest Element Require
7. test copy paste
    7.1. 5.4.1 DocumentRetrieveRequest Element Requirements
8. Dairy Comp page(VAS) with duplicated content (cloned with children) (cloned with children)
9. vas duplicate
10. Testing diacritics
11. CrossRiver test page 3
12. Test folder
    12.1. testing missing TOC header
13. CrossRiver test page
14. CrossRiver test page 2
    14.1. Untitled
15. retete
16. Test
17. Ashbyk img alt text dissapear
    17.1. Untitled
18. this is a h1 title
19. Baguette parce que car cela ouais
    19.1. Test doc
20. Indenting
    20.1. Another page
        20.1.1. Advanced filtering
    20.2. New page

21. test page Ocient
22. Reminder
    22.1. Toxic Language
    22.2. ASDASDSDA
    22.3. Test numbering on images issue (cloned)
        22.3.1. asdsadas
        22.3.2. Toxic Language
    22.4. asdsadasdas (cloned)
23. Virtual Account Number API Specification
    23.1. Create an Automatcher (Receivables) Account
    23.2. Get Account Status By Bank Account
    23.3. Batch Create
    23.4. Set Account Status
    23.5. Batch Status
    23.6. Get Account Status By ClientUniqueId
    23.7. securitySchemes
    23.8. schemas
    23.9. schemas
    23.10. schemas
    23.11. schemas
    23.12. schemas
    23.13. schemas
24. Judopay Transaction API
    24.1. Registering Cards
        24.1.1. /transactions/savecard
    24.2. Pre-Authorization
        24.2.1. /transactions/checkcard
        24.2.2. /transactions/registercard
        24.2.3. /transactions/preauths

24.2.4. /transactions/incrementalAuth
24.2.5. /transactions/collections
24.2.6. /transactions/voids
24.3. Payments
24.3.1. /transactions/payments
24.3.2. /transactions/refunds
24.4. 3DSecure
24.4.1. /transactions/{receiptId}/resume3ds
24.4.2. /transactions/{receiptId}/complete3ds
24.5. Receipts
24.5.1. /transactions/{receiptId}
24.5.2. /transactions
24.5.3. /transactions/{transactionType}
24.6. Payment Sessions
24.6.1. Create a generic payment session
24.6.2. Cancel an open payment session
24.6.3. Retrieve details of a completed payment session
24.6.4. Return a list of payment sessions
24.6.5. Retrieve the details of a single payment session
24.6.6. Retrieve the details of a single payment session for a payment
24.6.7. Retrieve the details of a single payment session for a preauth
24.6.8. Create a session for check card
24.6.9. Create a session for a payment
24.6.10. Create a session for a preauth
24.7. Other
24.7.1. /info
24.8. Models

24.8.1. Model: allowIncrementRequest
24.8.2. Model: networkTokenRequest
24.8.3. Model: applePayRequest
24.8.4. Model: cardAddressCommonAttributes
24.8.5. Model: cardAddressRequestAttributes
24.8.6. Model: cardAddressRequest
24.8.7. Model: cardAddressResponseAttributes
24.8.8. Model: cardAddressWebPaymentRequest
24.8.9. Model: cardDetailsCv2NotRequiredRequest
24.8.10. Model: cardDetailsCv2RequiredRequest
24.8.11. Model: cardDetailsStartDateRequest
24.8.12. Model: cardDetailsResponse
24.8.13. Model: cardDetailsListResponse
24.8.14. Model: cardTokenPreAuthPaymentRequest
24.8.15. Model: challengeRequestIndicatorRequest
24.8.16. Model: checkCardRequestPan
24.8.17. Model: collectionRequest
24.8.18. Model: complete3ds2Request
24.8.19. Model: consumerResponse
24.8.20. Model: createPaymentSessionResponse
24.8.21. Model: deviceResponse
24.8.22. Model: errorResponse
24.8.23. Model: googlePayRequest
24.8.24. Model: incrementAuthRequest
24.8.25. Model: infoResponse
24.8.26. Model: listPaymentSessionsResponse
24.8.27. Model: listTransactionsResponse
24.8.28. Model: paymentRequestApplePay
24.8.29. Model: paymentRequestCardToken

24.8.30. Model: paymentRequestGooglePay
24.8.31. Model: paymentRequestPan
24.8.32. Model: paymentSessionHistoricResponse
24.8.33. Model: paymentSessionHistoricListResponse
24.8.34. Model: paymentSessionRequest
24.8.35. Model: paymentSessionRequestForPreAuths
24.8.36. Model: paymentSessionRequestCommon
24.8.37. Model: paymentSessionResponse
24.8.38. Model: paymentSessionCancelResponse
24.8.39. Model: preAuthPaymentRequestCommon
24.8.40. Model: preAuthRequestApplePay
24.8.41. Model: preAuthRequestCardToken
24.8.42. Model: preAuthRequestGooglePay
24.8.43. Model: preAuthRequestPan
24.8.44. Model: primaryAccountDetailsRequest
24.8.45. Model: recurringPaymentRequest
24.8.46. Model: refundRequest
24.8.47. Model: registerCardRequest
24.8.48. Model: checkCardRequest
24.8.49. Model: registerCardRequestPan
24.8.50. Model: resume3dsRequest
24.8.51. Model: riskParametersResponse
24.8.52. Model: networkTokenisationDetailsResponse
24.8.53. Model: saveCardRequestCommon
24.8.54. Model: saveCardRequestPan
24.8.55. Model: scaExemptionRequest
24.8.56. Model: threeDSecureChallengeRequest
24.8.57. Model: threeDSecureCompletedResponse
24.8.58. Model: threeDSecureTwoChallengeRequiredResponse

24.8.59. Model: threeDSecureTwoDeviceDetailsRequiredResponse
24.8.60. Model: threeDSecureTwoCheckCardRequest
24.8.61. Model: threeDSecureTwoRequest
24.8.62. Model: transactionReceiptResponse
24.8.63. Model: transactionReceiptHistoricResponse
24.8.64. Model: transactionReceiptListResponse
24.8.65. Model: voidRequest

1. About

test

We couldn't verify the security of your connection

Access to this content has been restricted. Contact your internet service provider for help.

2. Untitled

3. Test Adobe cursor jumping on tables

Segment 1: Intelligent Asset Management Foundation

Duration: 25 minutes

This segment explores how AEM Assets combines folder organization, AI-powered metadata generation, and Workfront workflow metadata to create a searchable, governed asset repository. You will experience AI search capabilities including natural language queries, semantic search, and contextual search, and learn how to prepare assets for distribution.

Exercise 1.1: Asset Organization and Folder Hierarchy

Duration: 4 minutes

Understanding Folder Structure in AEM Assets

AEM Assets uses a structured folder hierarchy that mirrors your organizational needs, whether by brand, campaign, region, or asset type. This structure provides governance and permissions while AI makes everything searchable regardless of where assets reside. The folder hierarchy enables role-based access control and creates a foundation for both manual browsing and AI-powered discovery.

Value for Your Organization:

- Folder hierarchy enables role-based access control across teams
- Consistent structure across campaigns simplifies navigation for all users
- Supports compliance requirements with clear asset ownership

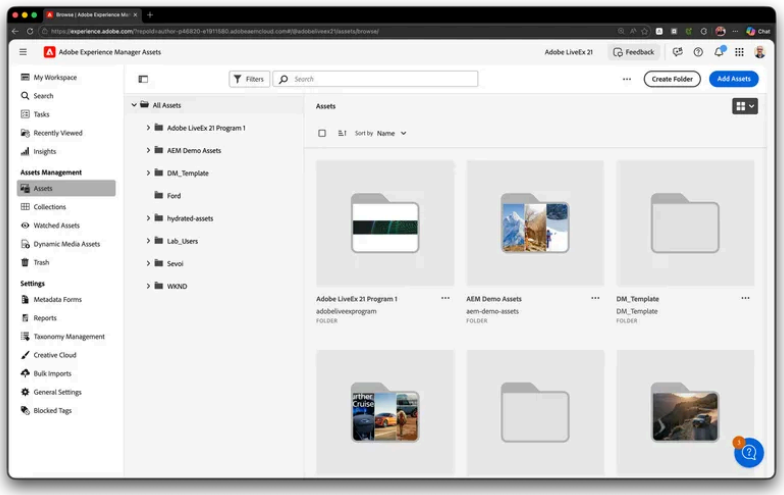
Reference Data: Folder Structure

Level	Example Path	Purpose
Brand	/content/dam/sevoi	Brand-level asset repository
Campaign	/content/dam/sevoi/campaigns/escape-beyond-ordinary	Campaign-specific assets
Region	/content/dam/sevoi/campaigns/escape-beyond-ordinary/apac	Regional variations
Asset Type	/content/dam/sevoi/campaigns/escape-beyond-ordinary/apac/photography	Content categorization

Step-by-Step Instructions

Step 1: Open AEM Assets

- Log into your AEM as a Cloud Service environment
- Navigate to the Assets view using the left navigation
- The Assets view provides a streamlined, marketing-focused interface



Click to Preview

Step 2: Navigate to the Sevoi Resorts brand folder

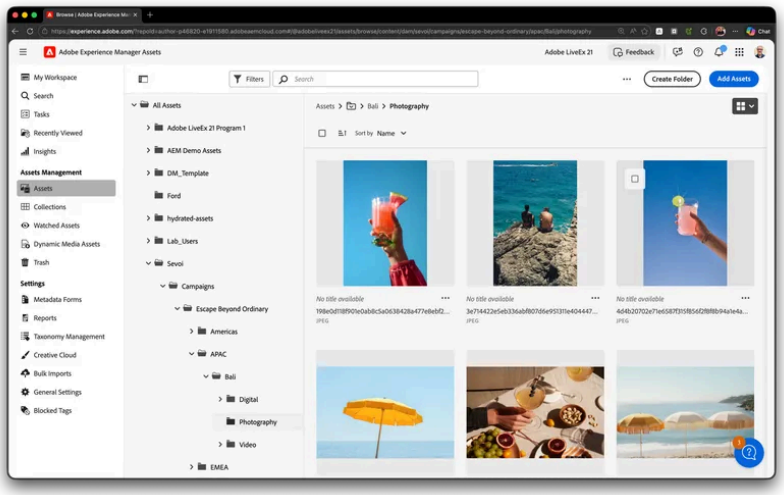
- In the Assets view, locate the content/dam folder structure
- Click on the Sevoi brand folder
- This is the brand-level repository for all Sevoi Resorts assets

Step 3: Expand the campaigns folder

- Click into the campaigns folder
- Locate the "escape-beyond-ordinary" campaign folder
- Click to open this campaign's asset repository

Step 4: Review the folder structure

- Observe the regional folders: APAC, EMEA, Americas, LATAM
- Note how each region has consistent subfolders
- This structure supports governance while enabling regional autonomy



Click to Preview

Step 5: Navigate to the APAC region folder

- Click on the APAC folder
- This contains assets targeted for the Asia-Pacific region
- Regional teams have appropriate access to their designated folders

Step 6: Observe the asset type subfolders

- Note the photography and banners subfolders
- This categorization helps teams find specific asset types quickly

Checkpoint: Can you navigate to the Bali campaign photography folder within the APAC region?

Governance Benefit: The folder hierarchy creates natural permission boundaries. Regional marketing teams can access their market's assets while

```
brand governance teams maintain oversight of the entire structure. This
reduces the risk of off-brand content usage while enabling team autonomy.
```

Exercise 1.2: AI-Generated Metadata and Smart Crops

Duration: 6 minutes

Understanding AI-Powered Asset Intelligence

When assets arrive in AEM Assets as a Cloud Service, Adobe AI immediately goes to work. It generates three distinct metadata fields automatically including a **Generated Title**, a **Generated Description**, and **Generated Keywords** which appear in a dedicated **AI Generated tab** on the asset's Properties page. These are separate from any manually entered metadata; if no manual title is defined, AEM uses the AI-generated title by default across card and list views.

Beyond metadata and tagging, **Smart Crop** which is powered by Adobe Sensei via Dynamic Media identifies the focal point of each image and pre-generates renditions for common aspect ratios automatically. This means channel-ready variations are available without generating separate stored files for each format.

Value for Your Organization:

- AI-generated metadata provide broader taxonomy coverage for discovery and filtering
- Smart Crops eliminate repetitive production work for common channel formats
- Assets are immediately usable across channels without designer intervention

Reference Data: AI-Generated Metadata Fields

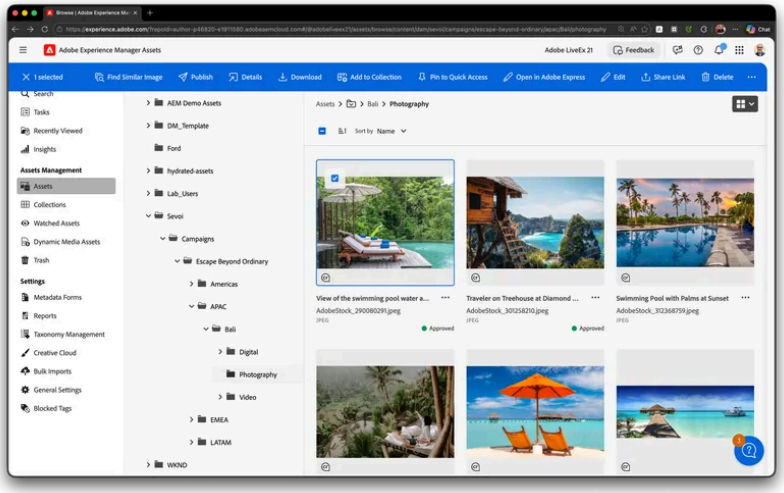
Field	Example Value	Where It Appears
Generated Title	Tropical Sunset Pool	AI Generated tab in asset Properties
Generated Description	Palm trees silhouetted against sunset reflecting in infinity pool at luxury resort	AI Generated tab in asset Properties
Generated Keywords	Resort, Sunset, Palm, Pool, Tropical, Luxury, Travel, Vacation	AI Generated tab in asset Properties
Smart Crop	2:1 (banner), 1:1 (social square), 4:5 (portrait social), 5:8 (story)	Dynamic Media Smart Crop renditions

Note on Smart Crops and Dynamic Renditions: Smart Crop renditions are distinct from static renditions like thumbnails. Static renditions are pre-generated files stored in the DAM. Smart Crop renditions are delivered dynamically and the crop coordinates are stored and applied at delivery time, meaning one source file serves all channel formats without generating separate stored copies.

Step-by-Step Instructions

Step 1: Select a Bali resort photography asset

- Navigate to the photography folder within the Bali campaign
- Click on one of the high-resolution resort images
- Select an image with strong visual elements (sunset, pool, architecture)



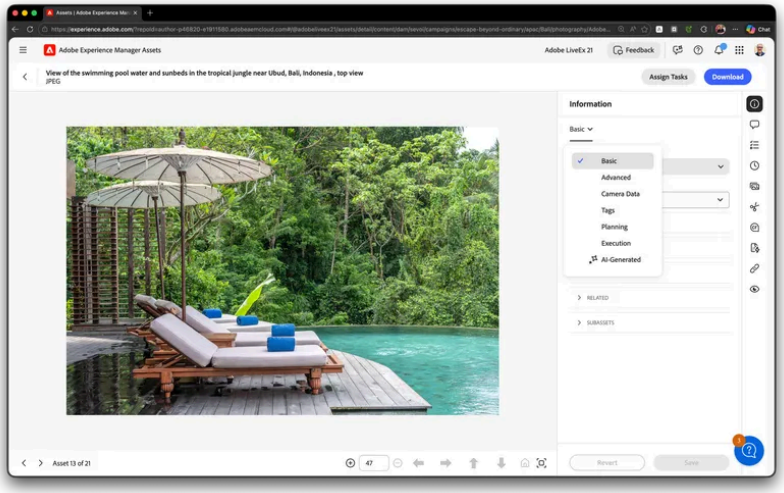
Click to Preview

Step 2: Open the asset details panel

- With the asset selected, the details panel should appear in the top menubar
- This panel shows all metadata associated with the asset

Step 3: Locate the AI Generated metadata tab

- In asset Properties, look for the AI Generated tab
- This tab is separate from the Basic tab where manual metadata lives
- The fields here were populated automatically when the asset was uploaded



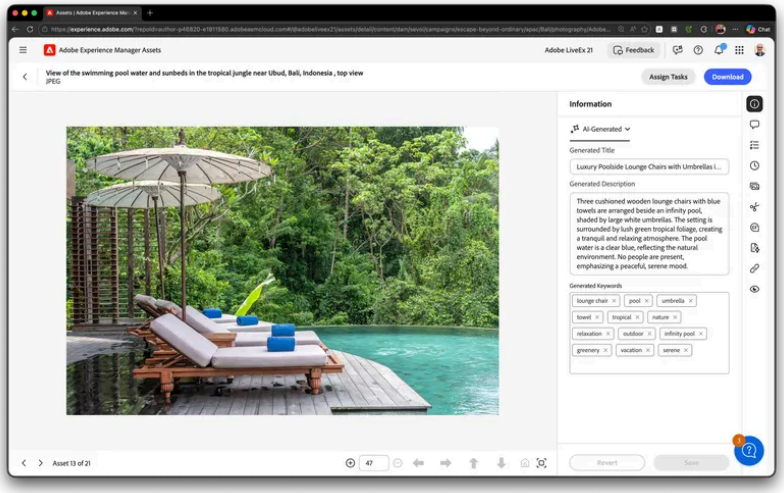
Click to Preview

Step 4: Review the Generated Title and Generated Description

- Read the AI-generated title which should describe the core subject of the image clearly
- Review the description for scene accuracy and detail

Step 5: Examine the Generated Keywords

- Find the Generated Keywords field in the AI Generated tab
- Review the list of AI-generated terms representing the asset's main themes
- These keywords are indexed for search alongside Smart Tags
- You can add or modify keywords manually if any relevant terms are missing



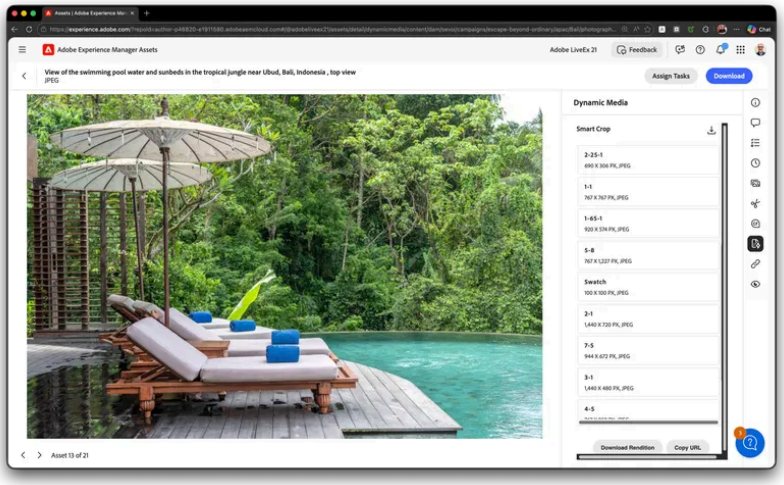
Click to Preview

Step 6: Click on the Smart Crops or Renditions section

- Locate the Renditions panel via the Dynamic Media icon in the right pane
- Look for the Smart Crop renditions within the list

Step 7: Preview each pre-generated Smart Crop

- Click through the available Smart Crop renditions:
  - 2:1 for web banners
  - 1:1 for social media squares
  - 4:5 for portrait social (Instagram feed)
  - 5:8 for Stories format



Click to Preview

Step 8: Observe focal point consistency

- Compare how the main subject remains visible and centered across all crops
- Each crop maintains the visual impact of the original without manual intervention
- If needed, click Edit Smart Crop to adjust the focal point manually for any ratio

Checkpoint: Can you see the Smart Crop variations for your selected image? Do all crops maintain the focal point? Can you locate the Generated Title, Description, and Keywords in the AI Generated tab?

Use Case Example: A global hospitality brand uploads 500 images from a property photoshoot. AI has generates searchable metadata for every image and created 2,000 pre-cropped dynamic renditions (4 crops x 500 images). What would have taken a production team a week happens automatically, freeing creative resources for strategic work.

Exercise 1.3: Workfront Metadata Integration

Duration: 4 minutes

Understanding Workflow-Sourced Metadata

AI can tell us what is in an image, but it cannot tell us why it was created. That context comes from Workfront. When assets sync from the creative workflow via the native AEM-Workfront integration, they bring campaign metadata, brand information, regional targeting, and launch dates. This combination of AI-generated and workflow-sourced metadata creates a complete picture that powers governance, reporting, and downstream activation.

Value for Your Organization:

- Campaign context enables reporting and ROI measurement
- Launch dates support content lifecycle management and expiration
- Single source of truth from Workfront eliminates data silos

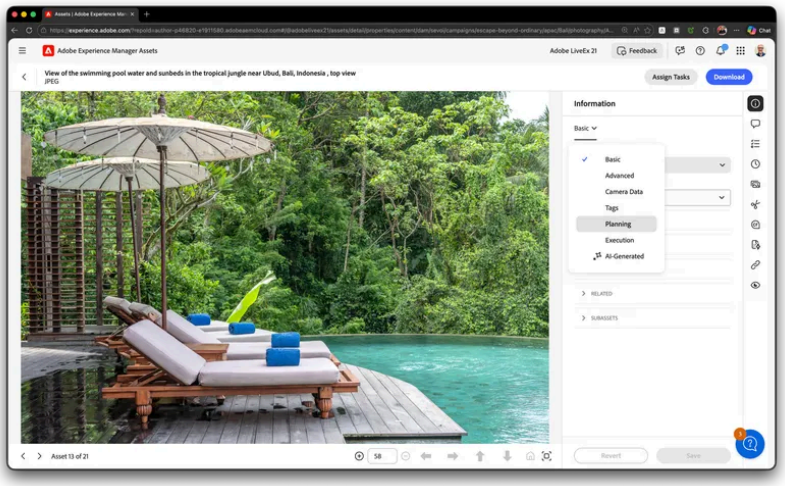
Reference Data: Enriched Metadata from Workfront

Field	Value	Source
Portfolio Name	Hands-On Lab	Workfront Project
Program Name	Customer Lab	Workfront Project
Project Name	Ben - Escape Beyond Ordinary	Workfront Project
Project Reference Number	24345	Workfront Project
Campaign	Escape Beyond Ordinary	Workfront Planning
Brand	Sevoi	Workfront Planning
Launch Date	April 14, 2026	Workfront Planning
End Date	October 14, 2026	Workfront Planning

Step-by-Step Instructions

Step 1: Keep the asset details panel open

- With your previously selected asset still active
- Click on the Info button in the right sidebar
- Open the Tab for Planning



Click to Preview

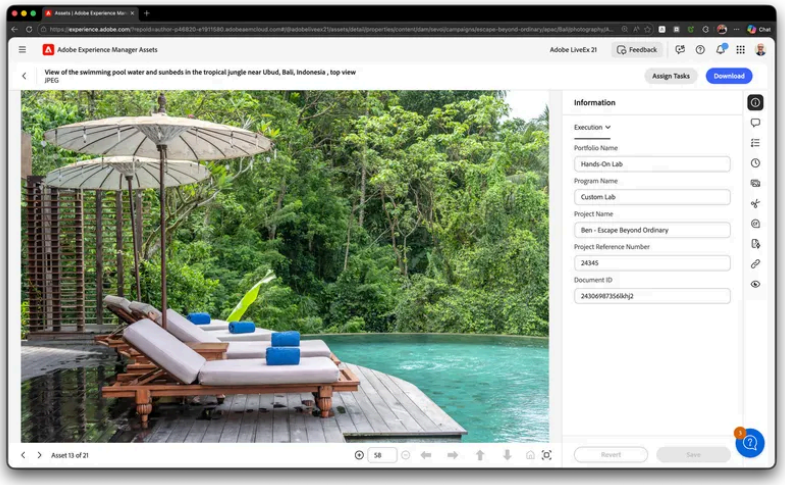
Step 2: Review the Campaign Planning fields

- Locate the Campaign field showing "Escape Beyond Ordinary"
- This metadata came automatically from the Workfront workflow
- Review the Brand field showing "Sevoi"

- Review the campaign activation timing fields

Step 3: Compare AI-Generated and Planning/Execution fields

- Scroll between the AI Generated tab and the Workfront-sourced fields
- Note how AI tells you "what" (sunset, pool, tropical)
- Note how Workfront tells you "why" (campaign, audience, timing)
- Together they create complete asset context



Click to Preview

Checkpoint: Can you see how the Workfront metadata adds business context to complement the visual metadata from AI?

Governance Benefit: With automated metadata enrichment, compliance teams can easily audit which assets are associated with which campaigns, verify brand usage across markets, and track asset utilization. When a campaign ends, the end date metadata can trigger automated review or archival workflows.

Exercise 1.4: AI Search Capabilities

Duration: 5 minutes

Understanding Intelligent Asset Discovery

AEM Assets provides multiple AI-powered search modes. **Omnisearch** (the standard search bar) factors Smart Tags and AI-generated metadata fields into results alongside standard metadata. **Contextual Search** allows users to type natural language text prompts

and AEM automatically converts them into search filters. Search results that match all terms in metadata fields display first, followed by results that match terms in Smart Tags only.

Value for Your Organization:

- Contextual Search converts natural language prompts into search filters automatically
- Global teams can search in their own language and terminology
- Semantic search surfaces relevant content even without exact keyword matches

Reference Data: AI Search Capabilities

Capability	Example	How It Works
Synonym Understanding	Search "ocean" finds assets tagged "beach," "sea," "coastline"	Conceptual matching via Smart Tags and generated metadata
Misspelling Tolerance	"trupical" returns "tropical" assets	Fuzzy matching in Omnisearch index
Multilingual Search	Spanish/German query finds English-tagged assets	Natural language processing via Contextual Search
Natural Language	"pool umbrellas for summer social media"	Contextual Search converts prompt to filters

Step-by-Step Instructions

Step 1: Click on the search bar in AEM Assets

- Locate the search bar at the top of the Assets view
- Click to activate the search field
- AI Search is enabled by default

Step 2: Search for Poolside Umbrella

- Type: poolside umbrella
- Press Enter to execute the search

Step 3: Observe synonym matching results

- Review the search results that appear
- Notice assets related to pools, patio umbrellas, and outdoor lounging scenes
- AI understood "poolside umbrella" includes related concepts via Smart Tags and generated metadata

Step 4: Clear search and try a misspelling

- Clear the search field
- Type: trupical paradise (intentional misspelling)
- Press Enter to search
- Observe that results still show tropical/paradise content

Step 5: Try a natural language search using Contextual Search

- Clear the search field
- Type: pool umbrellas for summer social media
- Press Enter and review results

Step 6: Search in another language (Spanish)

- Clear the search field
- Type: sombrilla de piscina (Spanish for "pool umbrella")
- Press Enter and observe results

Step 7: Try German search

- Clear the search field
- Type: Poolschirm (German for "pool umbrella")
- Verify results show luxury resort content tagged in English

Checkpoint: Could you find relevant assets even with the misspelling? Did the multilingual search return expected results?

Use Case Example: A multinational consumer goods company operates in 40 countries. Previously, regional marketers could not find assets unless they knew the exact English tags. With multilingual AI Search, the Japan team searches in Japanese, the Brazil team searches in Portuguese, and everyone finds the same relevant assets.

Exercise 1.5: Collections for Internal Team Collaboration

Duration: 3 minutes

Understanding Collections in AEM Assets

Collections allow teams to organize and share curated sets of assets without moving files or disrupting folder governance. Unlike folders, a collection can include assets from different locations across the DAM. AEM supports three types: **Lightbox** (a personal, per-user collection), **Standard Collections** (manually curated and shareable), and **Smart Collections** (dynamically populated based on saved search criteria).

Value for Your Organization:

- Standard Collections can be shared with configurable view or edit permissions
- Smart Collections automatically surface the latest assets matching defined criteria
- Collections work across folder boundaries, enabling cross-campaign curation

Reference Data: Collection Types

Collection Type	Behavior	Best Use Case
Standard Collection	Manually curated, static, shareable	One-time project deliverables, curated asset sets
Smart Collection	Dynamic, auto-updates based on saved search query	Ongoing campaigns, approved regional asset pools

Step-by-Step Instructions

Step 1: Navigate to the Collections view

- In AEM Assets, locate the Collections option in the left navigation
- Click to open the Collections interface
- You will see existing collections organized by Standard and Smart Collection types

Step 2: Create a new Smart Collection

- Click the Create Collection button

- Select Smart Collection as the collection type
- Name it: APAC Campaign Approved Assets

Step 3: Add filter criteria

- In the smart collection configuration, apply the following filters:
  - Brand = Sevoi
  - Region = APAC
  - Status = Approved
- These criteria will dynamically include all matching assets from across the repository

Step 4: Save the collection

- Click Save to create the Smart Collection
- The collection will immediately populate with matching assets
- Note the count of assets included

Step 5: View the collection contents

- Click on your newly created collection
- Review all assets that match the criteria
- Note how assets from different folders appear together in one view

Step 6: Observe cross-folder aggregation

- Notice assets from both photography and banners folders appearing together
- Users do not need to know folder structure to find the right assets

Step 7: Test dynamic updating

- Navigate back to an asset folder
- Approve an additional APAC asset (if not already approved)

Step 8: Refresh the collection

- Return to your Smart Collection and refresh
- Verify the newly approved asset automatically appears

Checkpoint: Can you see how your Smart Collection automatically includes

```
approved APAC assets from multiple folders?
```

```
Use Case Example: A regional marketing manager in Singapore opens their "APAC
Approved" Smart Collection and immediately sees every approved asset tagged
for their region. When new assets are approved, they appear automatically in
a living, always-current asset portal.
```

Exercise 1.6: Asset Approval for Distribution

Duration: 3 minutes

Understanding Approval Status as Governance

Not all assets in the repository are ready for broad distribution. In AEM Assets as a Cloud Service, each asset can be assigned one of three review states via the **Status** dropdown in the Basic tab of asset Properties: **Approved**, **Rejected**, or **No Status** (the default). When setting an asset to Approved, an **Approval Target** field appears, allowing you to direct the asset to Content Hub, Delivery (Dynamic Media with OpenAPI), or both, depending on your environment.

Value for Your Organization:

- Precise governance over which assets reach external distribution channels
- Approval Target field directs assets to the correct downstream channel
- Visual indicators make approval status immediately clear

Reference Data: Approval Status

Status	Purpose	Access Level
Approved	Finalized, ready for distribution	Content Hub users + AEM Assets users (per Approval Target)
Rejected	Flagged as non-compliant or off-brand	AEM Assets internal users only
No Status	Work in progress or awaiting review (default)	AEM Assets internal users only

Note: When setting an asset to Approved, the Approval Target dropdown determines whether the asset is published to Content Hub, Dynamic Media with OpenAPI (Delivery), or both. The default target is determined by which services are enabled in your environment.

Step-by-Step Instructions

Step 1: Select a Bali resort photography asset

- Navigate to the campaign photography folder
- Select one of the high-quality resort images
- Choose an image you consider campaign-ready

Step 2: Open the asset properties panel

- With the asset selected, access the properties/details panel
- Navigate to the Basic tab

Step 3: Locate the Status field

- Scroll through the Basic tab to find the Status dropdown
- Note the current state (No Status by default)
- This field controls distribution eligibility

Step 4: Change status to Approved

- Click on the Status dropdown
- Select "Approved" from the options (Approved, Rejected, No Status)
- If the Approval Target field appears, select Content Hub, Delivery, or both based on your environment

Step 5: Save the properties

- Click Save or Apply to commit the change
- Wait for confirmation that the change was saved
- The asset is now approved for distribution to the selected target

Step 6: Note the visual indicator

- Observe the visual badge showing approved status on the asset card

- This makes status immediately visible when browsing
- Other users can quickly identify approved assets

Step 7: Select 2-3 additional campaign assets

- Hold Ctrl/Cmd and click to select multiple assets
- Choose assets that are campaign-ready

Step 8: Bulk approve the selected assets

- With multiple assets selected, access the Bulk Metadata Edit panel
- Change the Status field to "Approved" for all selected
- Save to apply the change to all assets simultaneously

Checkpoint: Can you see the approved status indicator on your selected assets?

Governance Benefit: Approval status creates a clear gate between work-in-progress content and distribution-ready assets. Creative teams can continue iterating without risk of premature usage. When assets receive approval, they automatically become available to broader audiences through Content Hub, enabling self-service while maintaining brand control.

4. MedScout issue with boling and colors

Example text where I'll apply bolding/color

Example text where I'll apply bolding/colors

Example text where I'll apply bolding/colors

Example text where I'll apply bolding/colors

Example text where I'll apply bolding/colors

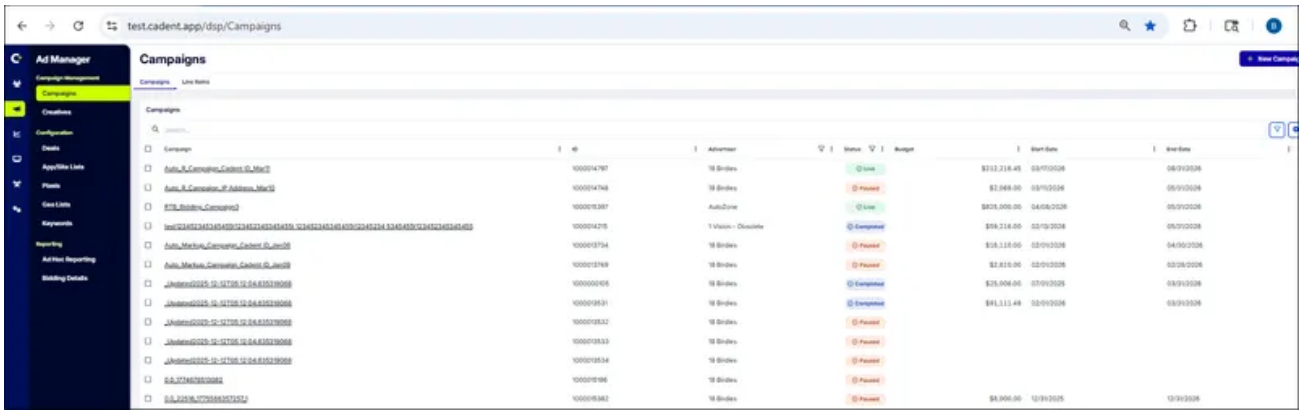
5. Cadent TOC issue

Release notes provide updates on new features and changes that occur on the Cadent Platform as part of a given release. We make new release notes available as soon as a new release occurs.

April 22,2026

Audiences page improvements

The Audience page has been updated to make the workflow better and help you be more efficient. New sorting and filtering options have been made available. In addition, you can download an Excel file that contains a list of all the content on the Audiences page. Set up the page to reflect your individual workflow and it will stay in place when you return to the page later. **Learn more.**



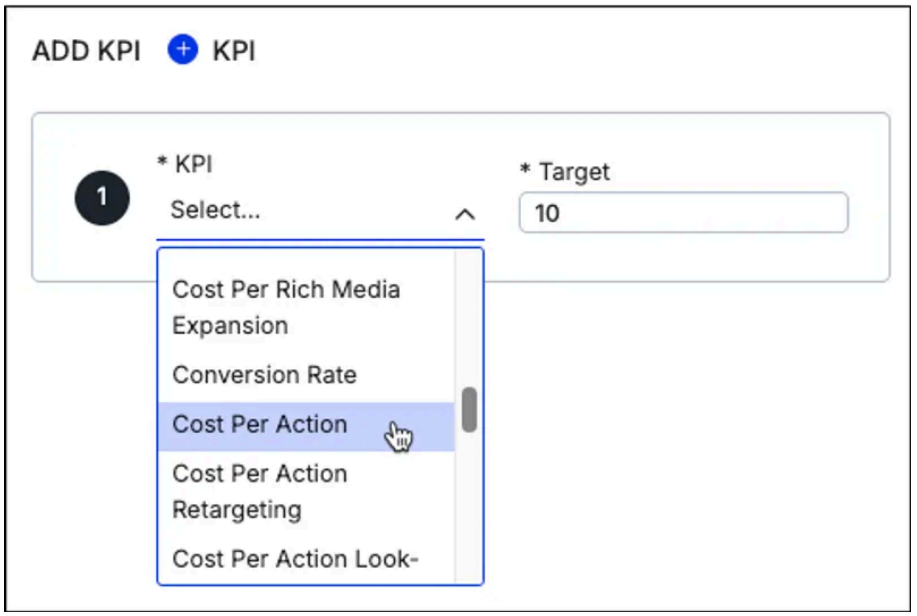
April 21, 2026

Ad Manager Feature

Cost Per Action KPI for line items

Ad Manager now supports cost per action (CPA) models, which are a campaign-specific models created automatically when you select **CPA** as the KPI for a line item, and then map the activities on the campaign.

CPA models are a key tool for driving strong KPI results by providing performance-based models trained on conversion data, like a sale or an app installation. This is different from other models, such as click models, where they optimize towards clicks regardless of whether they lead to a conversion.



Cost Per Action is a line item KPI

April 20, 2026

Ad Manager Features

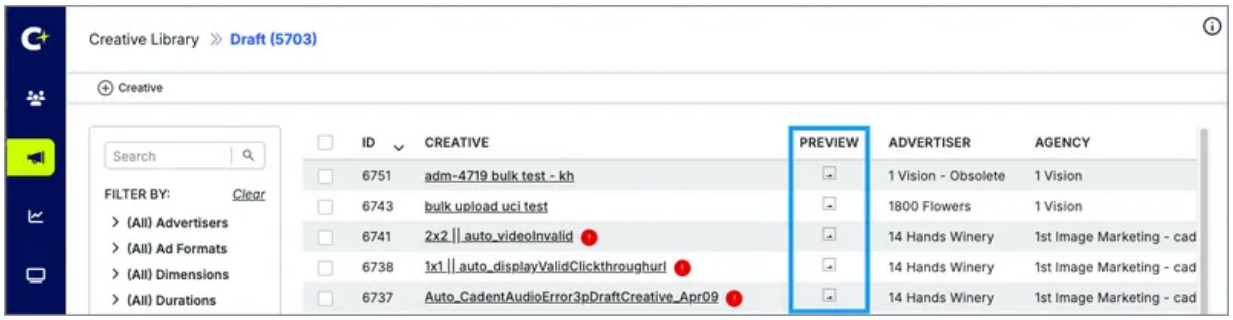
Preview and test your draft creatives

You can now preview and test your draft creatives. This quality control step ensures your draft creatives render and function exactly as intended before you **approve your draft creatives**.

Learn more about **previewing and testing creatives**.

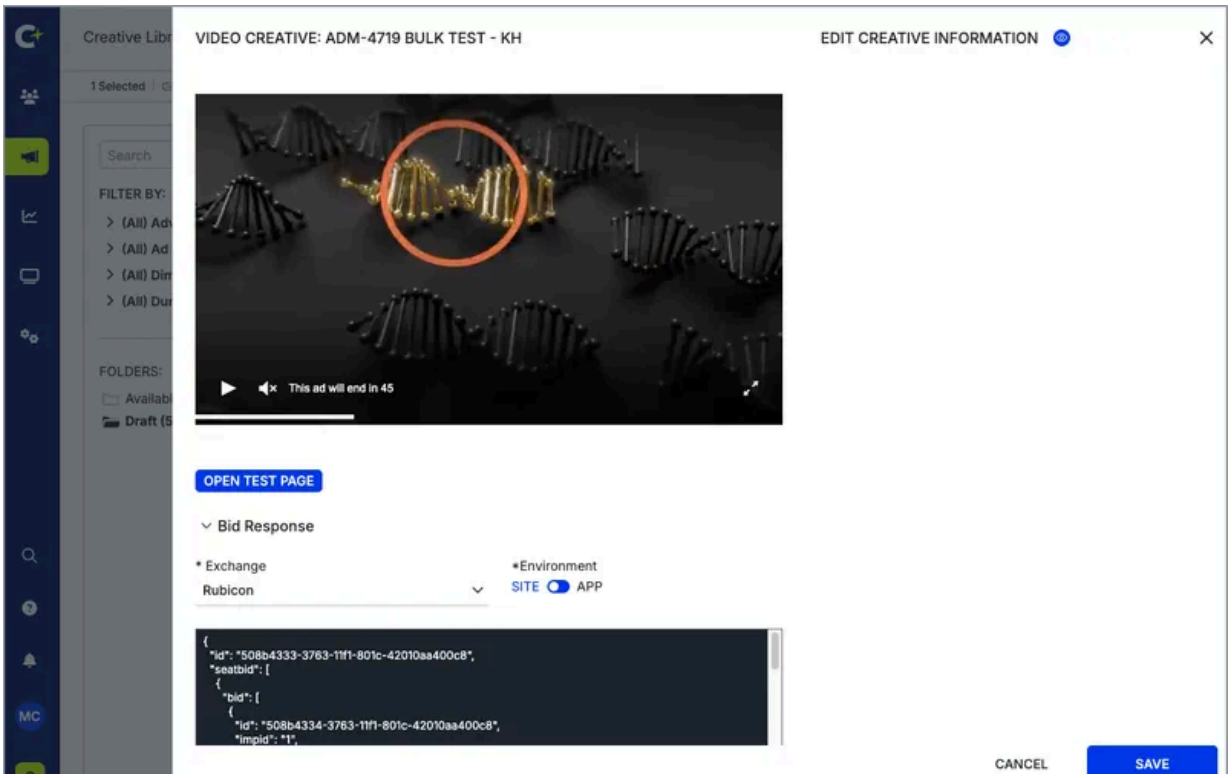
How to preview and test a creative

1. Navigate to the Creative Library, then click the **Preview** icon next to the creative. This opens the preview and test panel.



Preview icon in the Creative Library

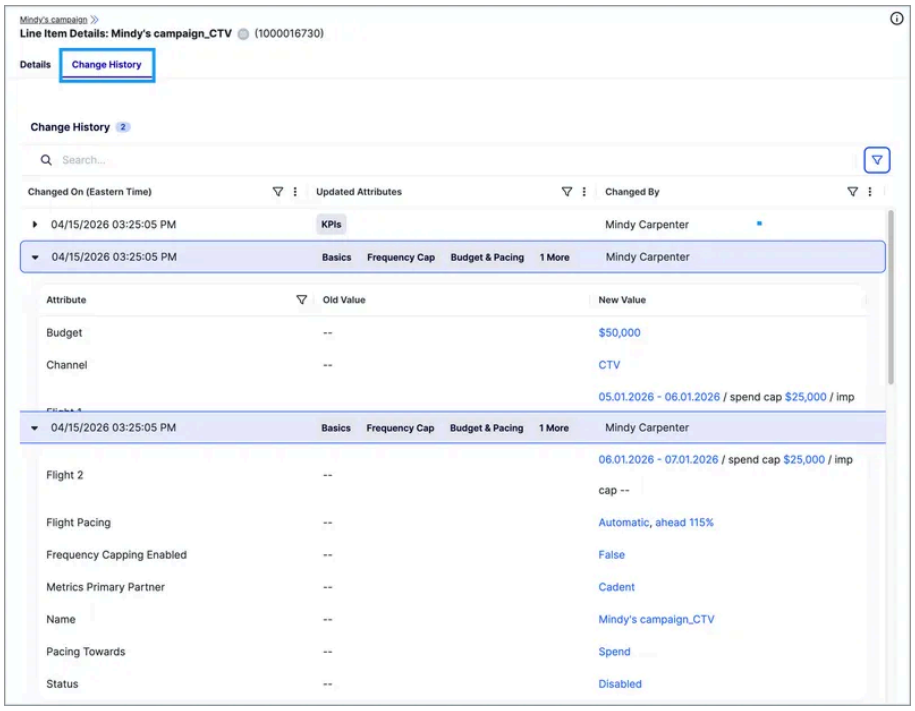
2. Here, you can do the following:
    - Click the creative preview to test the click-through URL.
    - View the bid response.



Introducing line item Change History log

You can now review a complete log of all changes made to your line items. Simply navigate to the new **Change History** tab on the Line Item Details page. There, you will find a detailed log of every modification, including:

- Date the change was made (Eastern Time)
- Attribute that was changed
- Old Value
- New Value
- Who made the change



Change History tab

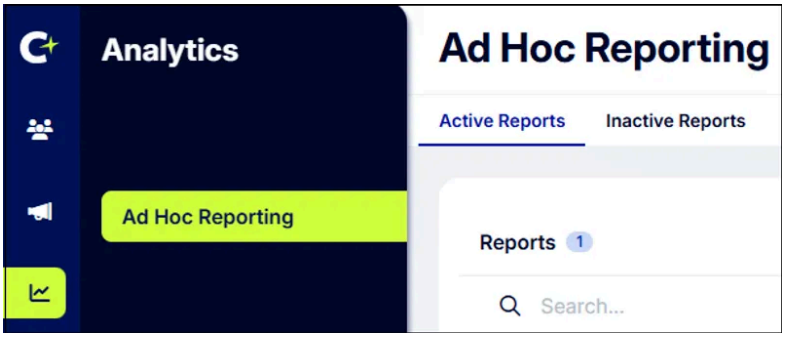
This feature gives you greater transparency and makes it easier to troubleshoot by providing a clear audit trail of all adjustments.

Learn more about the **Change history log**.

Analytics - Ad Hoc Reporting Features

Ad Hoc Reporting has moved

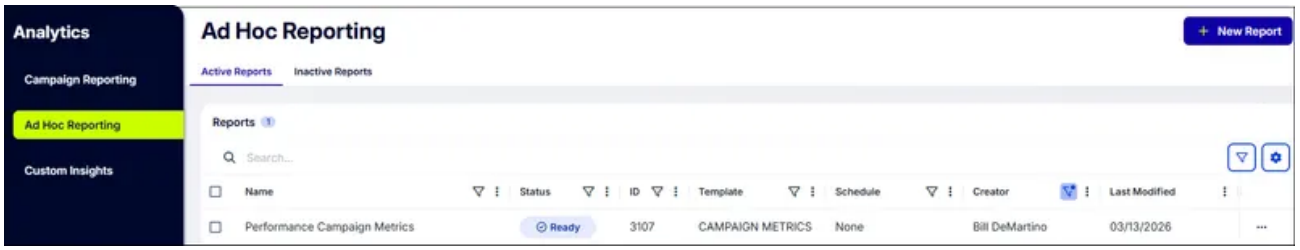
You can now find Ad Hoc Reporting by clicking the Analytics icon on the left-side nav bar. In the past, it was under Ad Manager.



Ad Hoc Reporting page improvements

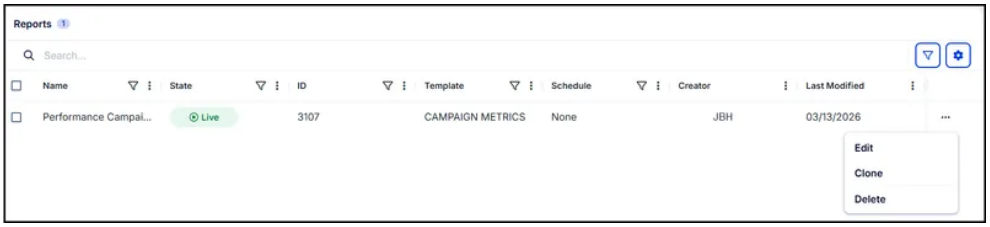
The Ad Hoc Reporting page has been updated to improve it's workflow and efficiency. You have the ability to add or remove columns. More powerful filters provide the

capability to view a specific set of reports and sort them to suit your needs. **Learn more.**



Clone reports for faster changes

On the Ad Hoc Reporting page, you can now clone a report. Cloning allows you to save time when creating a new report by using an existing report as the foundation for a new one. After you create the clone, you can then change details in the clone to meet the needs of new report. **Learn more.**



April 6, 2026

New bulk action: Update Optimization settings for multiple Ad Groups

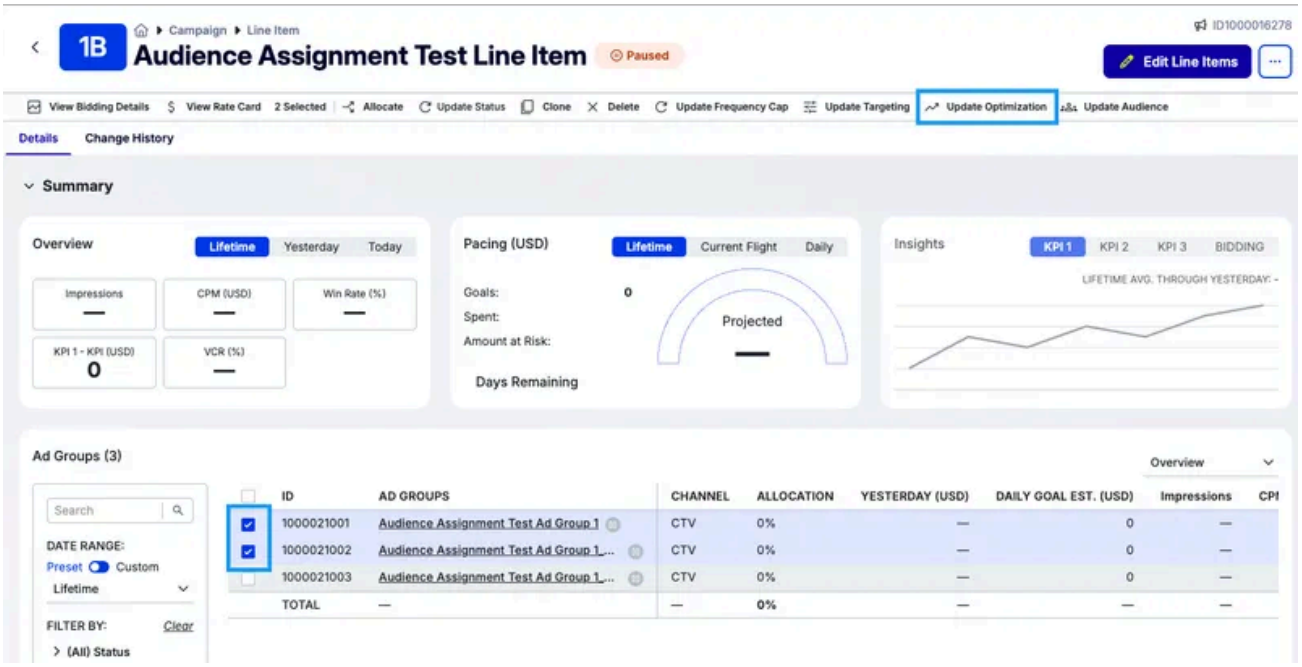
You can now apply optimization setting changes to multiple ad groups simultaneously, streamlining your campaign management workflow and ensuring consistency across your ad groups. Previously, updating these settings required editing each ad group one by one.

You'll find this bulk update option on the following pages:

- **Campaigns Details**: For the list of line items appearing there
- **Line Item Listing**: On the Campaigns page
- **Line Item Details Page**: For the list of ad groups appearing there

To bulk update ad group optimization:

1. Select one or more ad groups. Note that ad groups must be associated with the same advertiser to perform this bulk update.
2. Click **Update Optimization** to open the **Optimization** panel.
3. Click **Save** to apply your changes across all selected items.

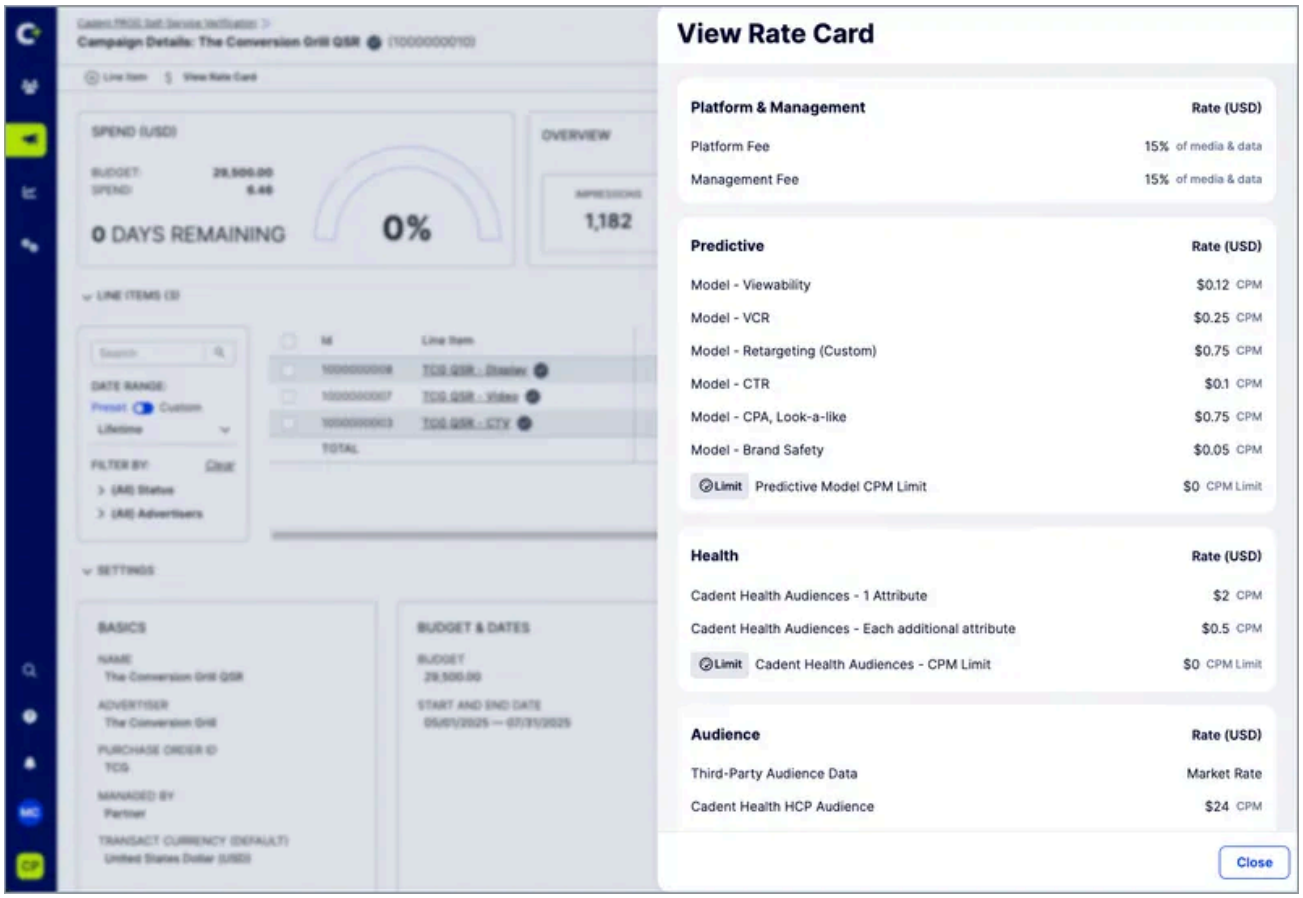


Easily reference your pricing with Rate Card Viewer

We're excited to announce the launch of the new Rate Card Viewer in Ad Manager. This new view provides quick, transparent access to your Rate Card, reducing time spent looking for your rates.

To access the Rate Card, click **View Rate Card** in one of the following screens:

- Campaign Details
- Line Item Details
- Ad Group Details



Rate card viewer

April 1, 2026

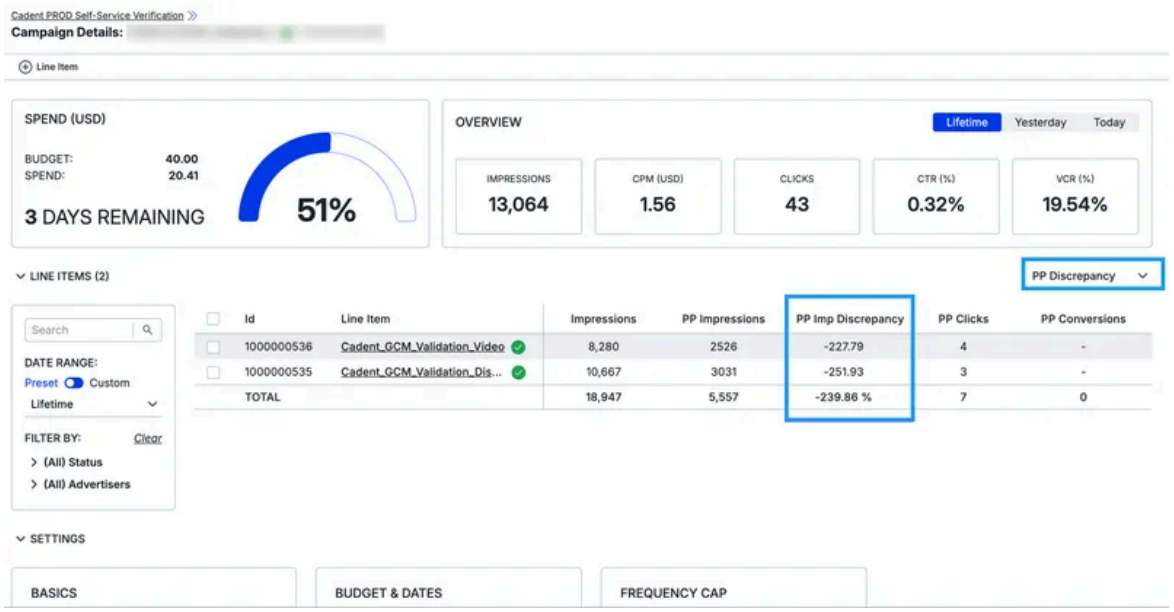
Ad Manager release notes

Resolved issue: Primary Partner Impression Discrepancy

The methodology used for calculating the difference between Cadent-tracked impressions and Primary Partner impressions (e.g., those tracked by an ad server like CM360) was displaying inaccurate numbers. We've resolved this issue, so now the **PP Imp Discrepancy** column is reflecting accurate numbers.

To view the PP Imp Discrepancy column:

1. Navigate to the **Campaign Details** or **Line Item Details** page.
2. Select **PP Discrepancy** from the view selector.



March 23, 2026

Ad Manager release notes

Report filters for Ad Hoc Reporting have been simplified

When creating a new report, you can now filter reports only by agency. No other filters are required. This makes it possible to produce more comprehensive reports than were previously possible. **Learn more.**

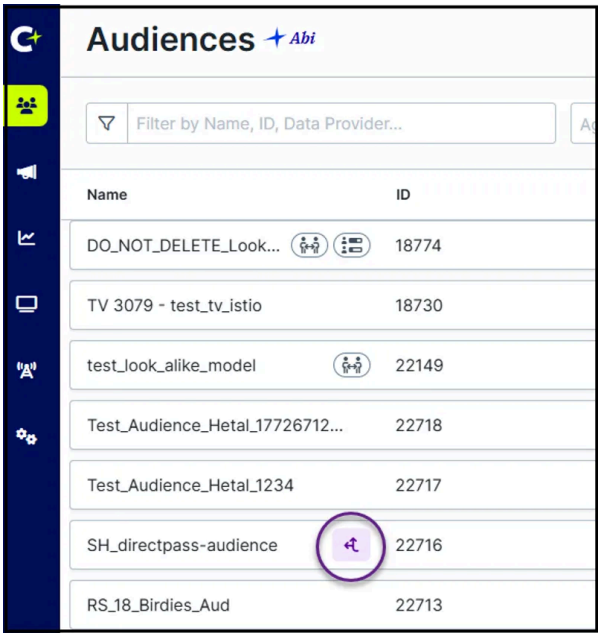
Audience Manager release notes

DirectPass icons added in Audience Manager

A new DirectPass icon now displays alongside DirectPass datasets in Audience Manager making them easier to spot. DirectPass uses a specialized dataset onboarding and deployment workflow that allows the direct ingestion and deployment of raw device identifiers (such as Device IDs and IP addresses) without first resolving them to Cadent's household graph. DirectPass datasets don't include the household level information.

DirectPass icons display in the following locations in Audience Manager:

- Audiences page
- Deployments page
- New Audience, My Datasets tab
- New Audience, Marketplace Datasets tab
- New Audience, Shared with Me tab


Audiences
Abi
Filter by Name, ID, Data Provider...
Name
ID
DO_NOT_DELETE_Look...
18774
TV 3079 - test_tv_istio
18730
test_look_alike_model
22149
Test_Audience_Hetal_17726712...
22718
Test_Audience_Hetal_1234
22717
SH_directpass-audience
22716
RS_18_Birdies_Aud
22713

DirectPass icon on the Audiences page

Audience tags display on the Audiences page

You can now see audience tags with audiences on the Audiences page. Tags provide the ability to create categories of audiences that share commonality. Although the tags do not display by default, you can add them to the page using the icon on the Audiences page.

The % Reach indicator on the New Audience page has been updated

On the**Total Reach** section of the New Audience page, the **% Reach** column has been renamed **Percentage Change** to improve clarity. This column displays the percentage change in total reach as you configure a new audience. It is updated in real time when you make changes to the audience you are building. Learn more

March 18, 2026

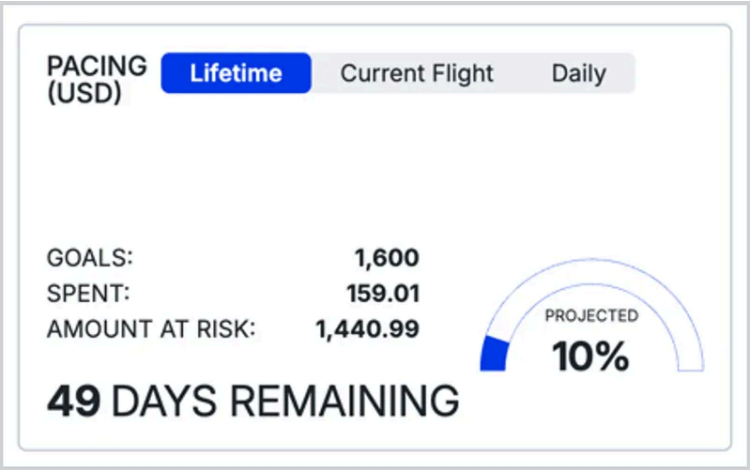
Ad Manager release notes

New 'Lifetime' view for line item pacing

We've added a new **Lifetime** timeframe to the line item Pacing tile. This allows you to see pacing metrics across all of a line item's flights, giving you a more holistic view of performance and helping you better understand and analyze the performance of your long-term campaigns.

Here's a breakdown of the metrics you'll see in this new view:

Metric	Description
Goal	This is the overall goal for the line item, depending on how your line item is pacing: • If pacing toward impressions, the Goal reflects Flight Impressions. • If pacing toward spend, the Goal reflects Flight Spend Cap.
Spent	The total amount spent through yesterday.
Amount at Risk	This shows the portion of the goal that's projected to be missed, giving you an at-a-glance view of pacing performance. Amount at Risk is calculated using this formula: (100% - Projected %) * Goal = Amount at Risk
Days Remaining	The number of days left in the current flight, not including today.



Simplified way to disable campaigns, line items, and ad groups

We've streamlined the process for disabling items in your account, making it faster and more intuitive to manage your campaigns.

Disabling campaigns and line items

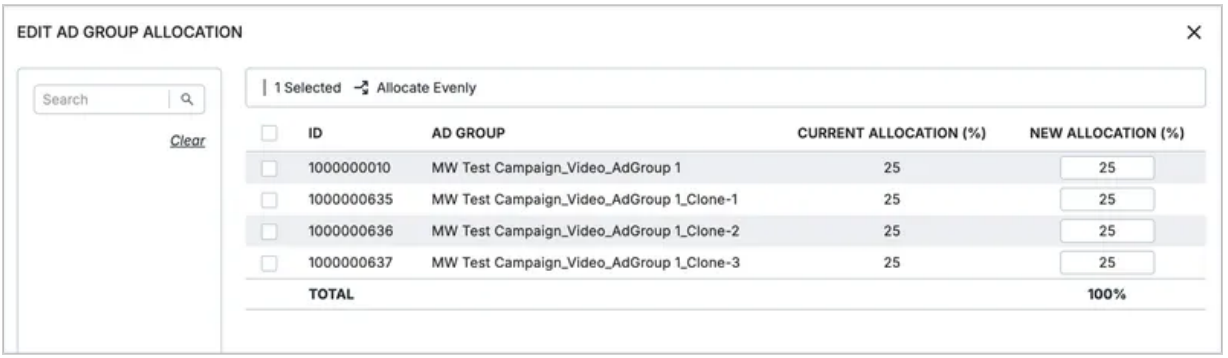
You can now directly set the status of an entire Campaign or Line Item to **Disabled** without changing its underlying budget setup. Previously, you had to go into each Ad Group individually to disable the allocation. This change makes pausing your campaigns a much quicker, one-step process.

Disabling ad groups

You can set an individual Ad Group to **Disabled**, but only if the parent Line Item's active budget allocation already totals 100%.

To check the budget allocation for a Line Item, simply follow these steps:

1. Go to the **Line Item Details** screen.
2. Select the **Ad Group** and click **Allocate**. This opens the Edit Ad Group Allocation panel.
3. Review the budget allocation details, and if necessary, make edits to bring the total allocation to 100%.



Save time with bulk pixel uploads

We're excited to announce a new feature that will make your workflow significantly more efficient. You can now create pixels in bulk—including universal tags, activities, and action activities—directly into Ad Manager, saving you from the manual effort of creating them one by one.

How to Upload Pixels in Bulk

Follow these steps to get started:

1. Navigate to the **Pixels** screen from the main menu.
2. Click the **+Pixels** button and select the **Bulk** option. This will open the Add Pixels in Bulk panel.

3. Select the **Advertiser** you want to associate the pixels with.
4. Download the **sample file** provided. This template ensures your data is formatted correctly for a successful upload. Add your pixel information to this file and save it to your computer. If you're trying to make any new activities for an existing Universal Tag, you should download a new template from the panel to ensure your existing Universal Tags are pre-populated in the dropdown list on the Activities tab of the template.
5. Drag and drop your completed file into the upload area, or click to browse for it on your computer.
6. Click **Add**.

Once the upload is complete, you will be automatically returned to the main Pixels screen where you'll see a success message confirming your new pixels have been added.

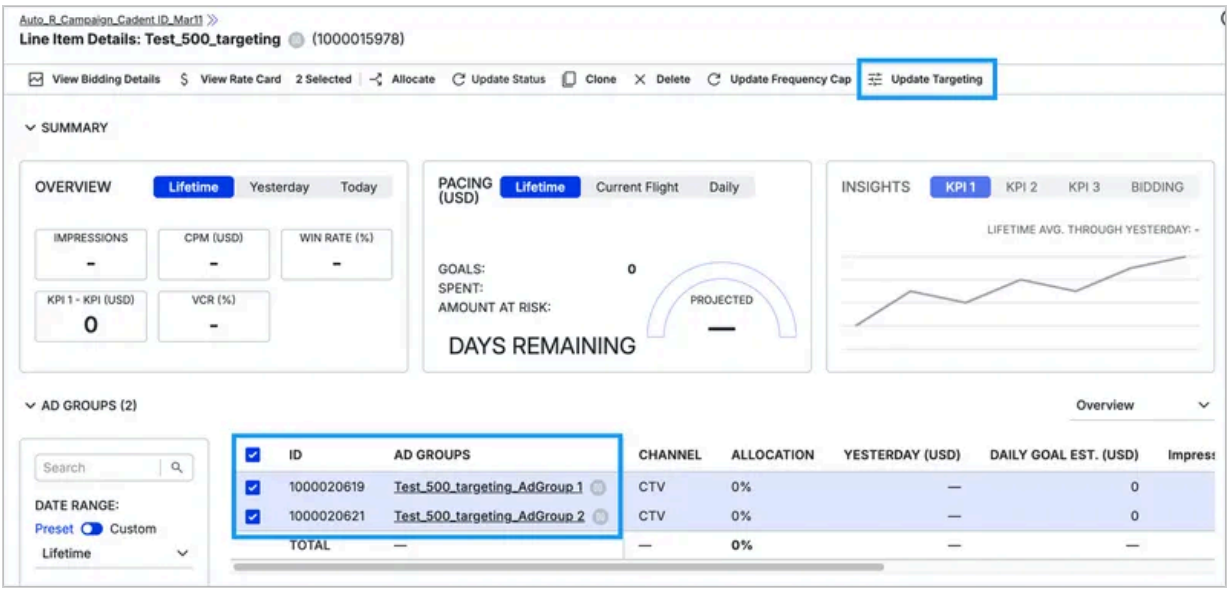
Streamline your targeting with bulk actions

You can now update **Custom Categories**, **Supply Categories**, and **Network & Channel** targeting for multiple ad groups at the same time.

> **Key Details to Remember**
>
> - **Custom Categories:** You can select from pre-made custom categories during a bulk update. New categories must be created beforehand.
> - **Network & Channel:** This targeting option is specifically for CTV.
> - **Supply Categories:** These are Jounce-defined supply categories.

How it works:

1. Navigate to the Line Items Details screen and select the ad group(s) you want to update, and then click **Update Targeting**. This opens the Add Targeting panel.



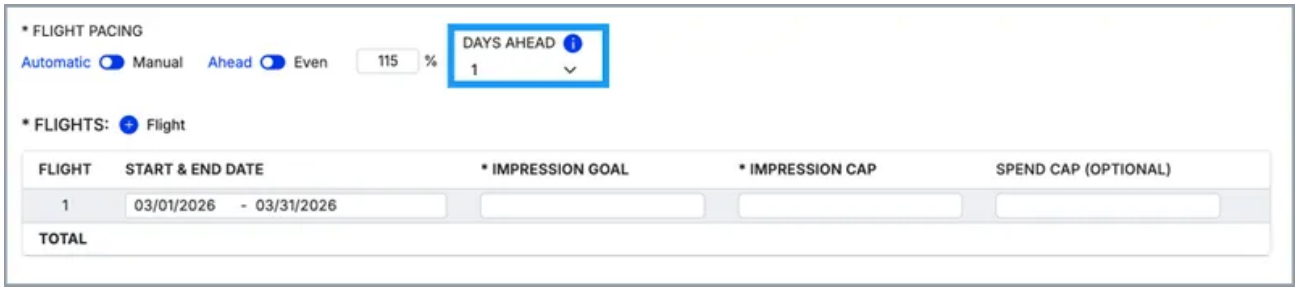
2. In the **Add Targeting** panel, make your new targeting selections and close the panel.
3. You're all set! The new targeting options will be reflected in the **Ad Group Details** screen for each Ad Group you updated.

February 19, 2026

Ad Manager release notes

Line item update: Days Ahead setting

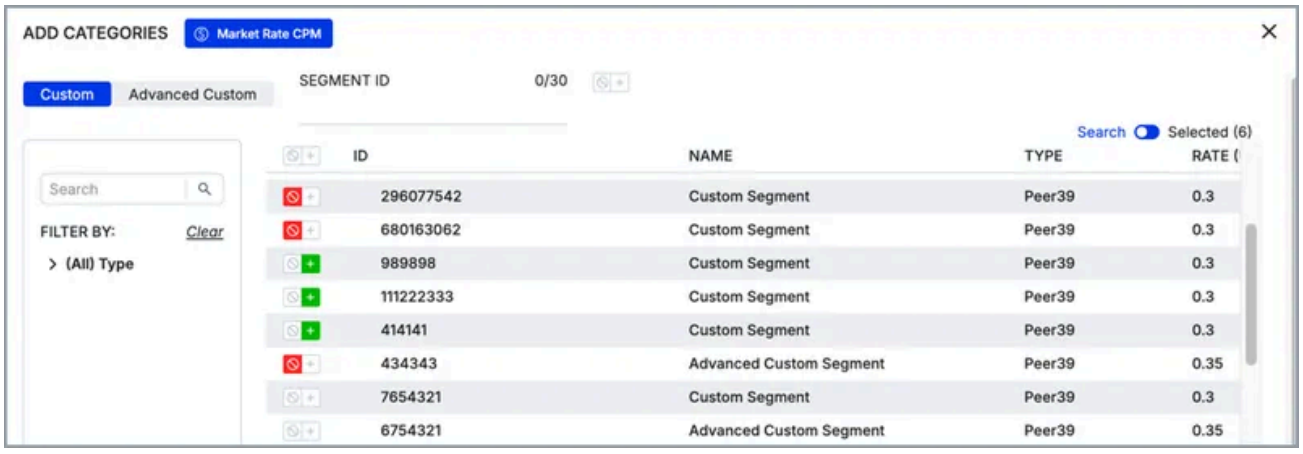
You now have the ability to indicate whether a line item should aim to deliver in full a certain numbers of days prior to the end date.



Targeting update: Categories targeting

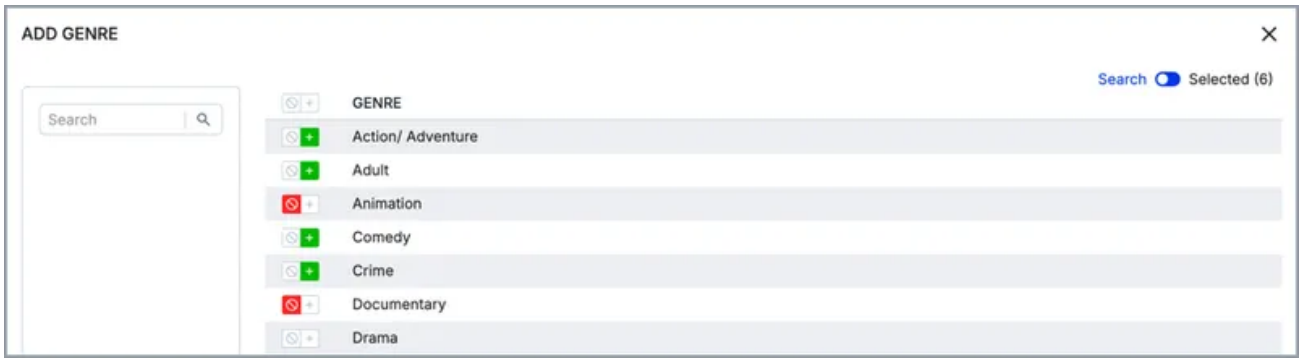
You can now target IAB or Peer39 content categories, including custom and advanced Peer39 segment creation and selection, to ensure your ads appear on relevant content.

You can target specific IAB categories or Peer39 segments, or you can exclude categories to prevent your ads from appearing on undesirable content.



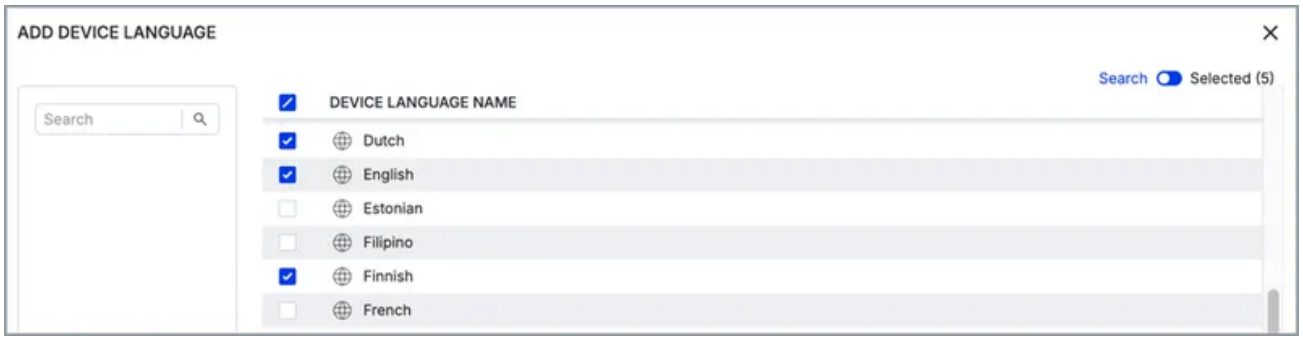
Targeting update: Genre targeting

Genre targeting allows you to align your campaigns with specific types of CTV video content. This feature enables you to target or exclude content based on the genre—for example, Comedy, Sports, Drama, Documentaries, and True Crime. You can either include specific genres to target or exclude genres you want to avoid.



Targeting update: Device Language targeting

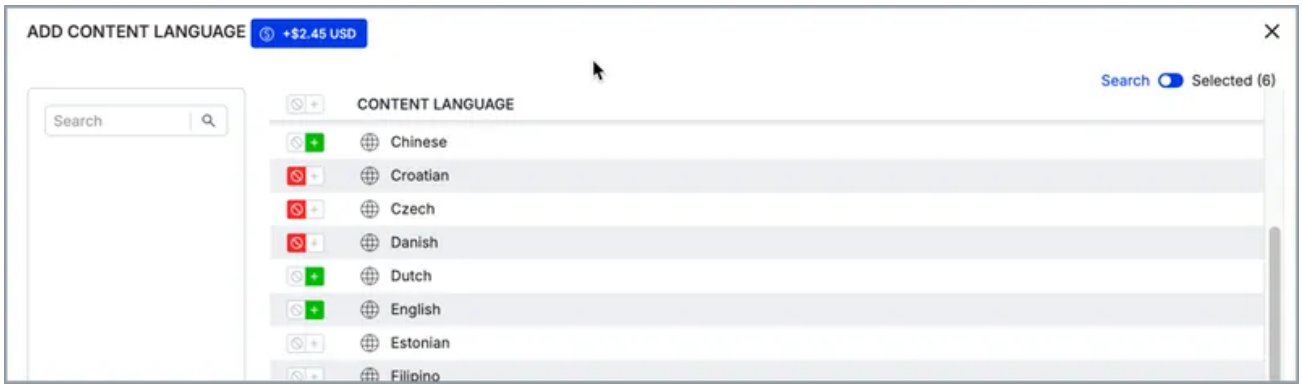
Device Language targeting lets you deliver ads specifically to users whose devices are set to a particular language. It's a straightforward way to ensure your message is relevant and understood by your target audience, especially in multilingual regions or global campaigns.



Targeting update: Content Language targeting

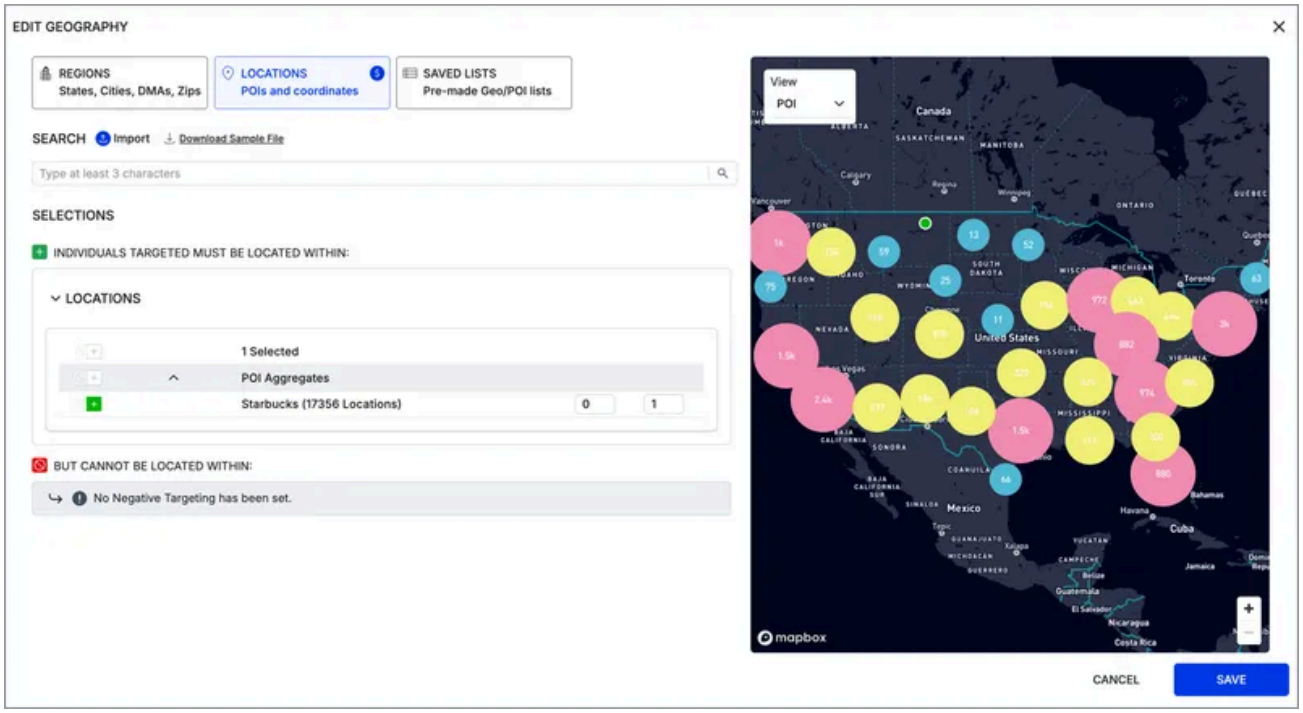
This method allows you to target viewers based on the primary language of the CTV content itself. For example, you can choose to have your ads appear only during Spanish-language programs, regardless of the viewer's device settings.

You can choose to either target or exclude language.



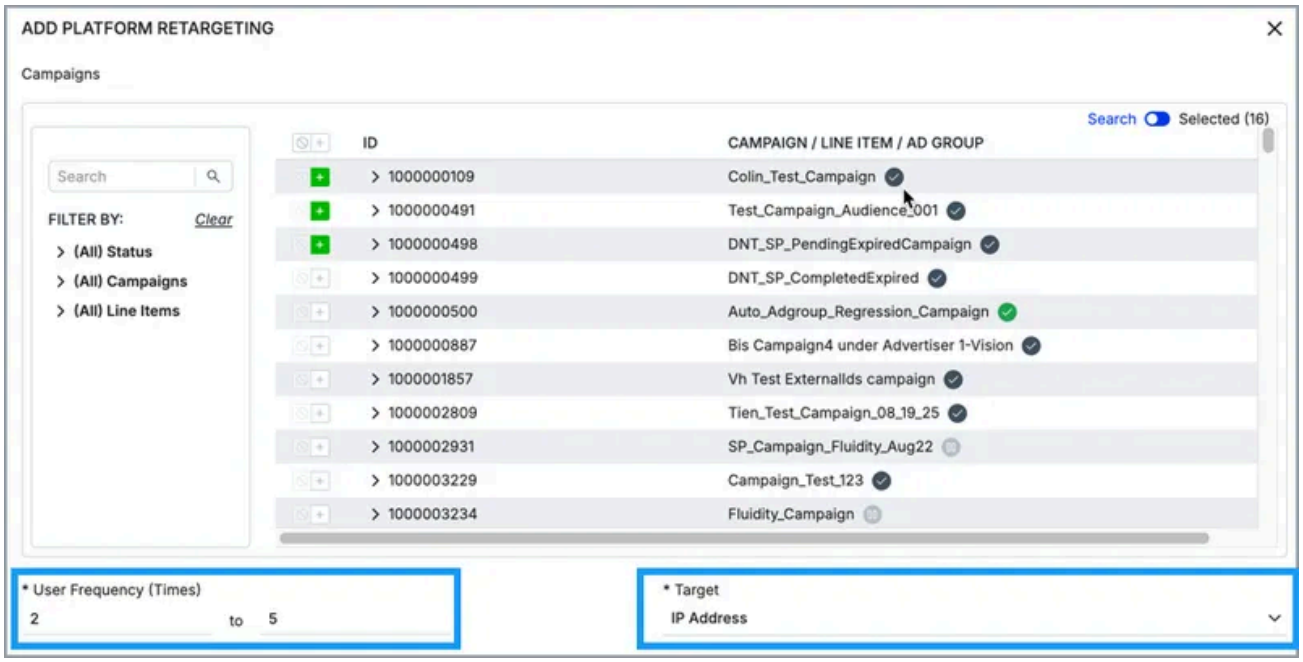
Targeting update: Geo POI targeting

You can target POIs via Geo targeting, which are presented to the user as an aggregate vs single point selection. For example, in one click you can target 4,000 Starbucks vs. having to select each one at a time.



Targeting update: Platform retargeting

Platform retargeting allows you to target users who have previously exposed to your campaigns.



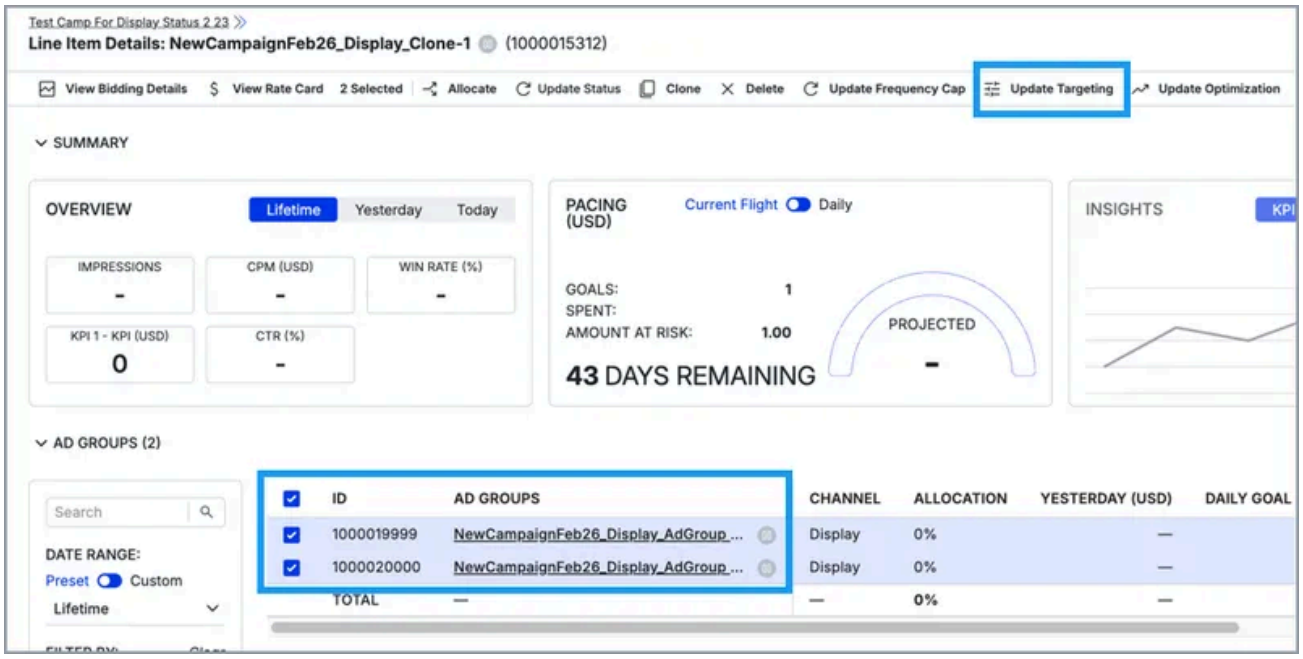
- You can choose to either positively target (include) or negatively target (suppress) the selected audience.
- User Frequency:This allows you to set a minimum and maximum range for how many times a person has been served your ad. For example, if you set a minimum of 2 and a maximum of 5, the system will only target users who have seen the ad at least 2 times but no more than 5 times.
- Target: This is the identifier used to count the frequency. You can select to track users by their IP Address, a browser Cookie, or All (which uses both methods).

Targeting update: Bulk actions

You can now select multiple ad groups in the line item details view to apply bulk targeting actions (see the screenshot below). The selections you make will override the previous selections.

The targeting options you can bulk update are:

- App/Sites list
- Packages
- Geography
- Deals
- Keywords



Reporting release notes

Fewer filters are required when creating an ad hoc report

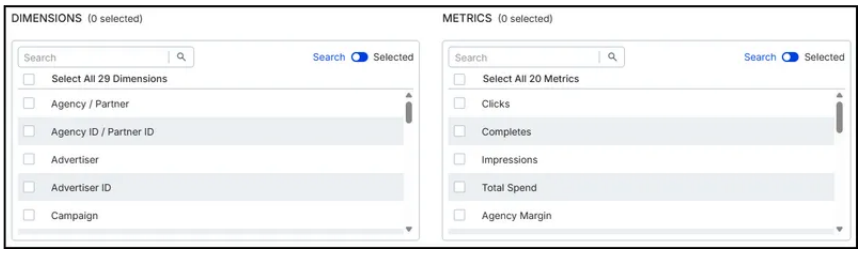
We have reduced the number of filters you need to apply when creating an Ad Hoc report. Now, you are only required to select an agency. This makes it possible to create broader reports than were possible in the past. Learn more about **Creating ad hoc reports**.

New reporting dimension added to Performance Insights templates

To provide you with a way to see more detail on your creatives in reports, a new reporting dimension, **Creative Type**, has been added. A creative type defines how a user interacts with a digital ad. For example, **video**, **display**, and **banner** are creative types that apply to online ads. The creative type dimension is being made available on the following report templates:

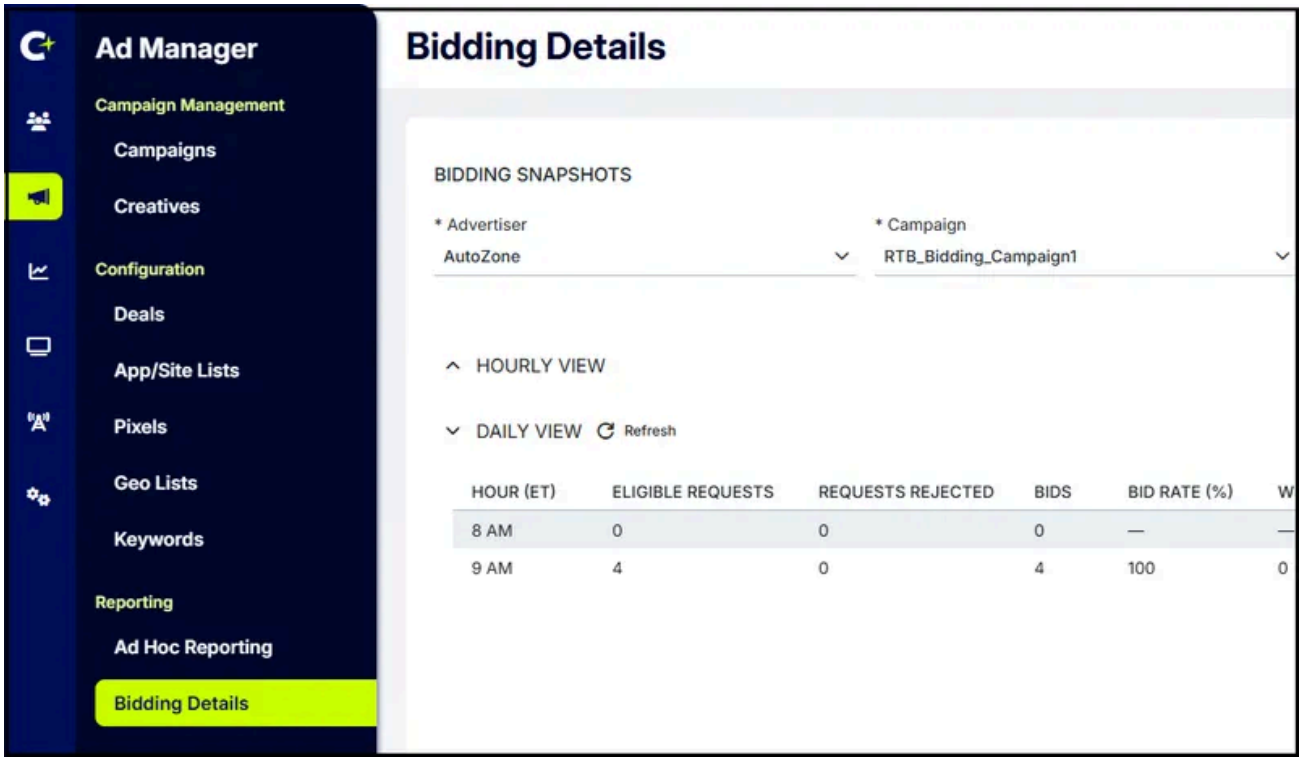
- Campaign metrics
- CTV metrics
- Video metrics
- Geo targeting metrics
- Reach and frequency
- HCP metrics

Learn more about **Performance Insights Templates**.



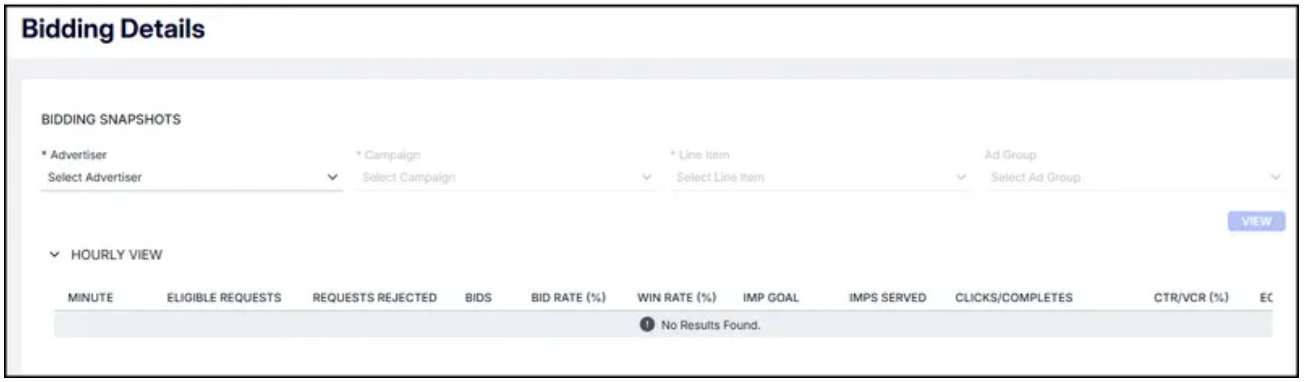
Bidding Snapshot renamed Bidding Details

When you access the Bidding Snapshot report, you will notice that it has been renamed **Bidding Details**. The new name displays on the left-side navigation bar and on the top of the page in Ad Manager. It is the same report as before but with two new additions described below.



Bidding Details report adds an hourly view

In addition to the Daily and Troubleshooting views, you now have the option to select an **Hourly View** of line items and ad group details. The hourly view displays the current hour's activity, and it can be refreshed to show the latest updates.



New fields display on Bidding Details report

The following new fields have been added to the Bidding Details report results for hourly and daily report results:

- **Spend Goal** - The monetary target an advertiser or media buyer aims to reach within the timeframe of a line item or ad group.
- **Total Spend** - The amount of money your campaign or line item has spent from the budget to win impressions.

February 4, 2026

Audience Manager release notes

DirectPass Data Ingestion

DirectPass uses a specialized dataset onboarding type and deployment workflow that allows the direct ingestion and deployment of raw device identifiers, such as Device IDs and IP addresses, without first resolving them to Cadent's household graph. DirectPass datasets don't include the household level information.

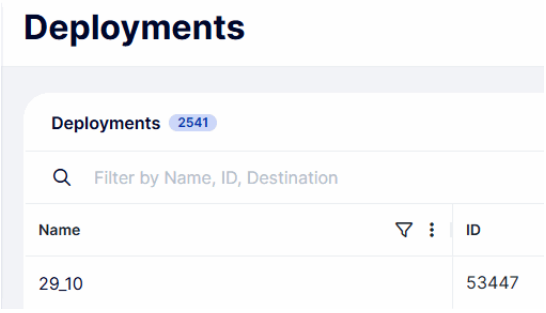
DirectPass enables specialized workflows that require the use of raw, unaltered identifiers. You can use DirectPass under the following conditions:

- Targeting for regions outside the United States, such as Canadian segments for use in AdManager.
- Passing raw identifiers directly to destination partners that have their own identity resolution systems.

Note: You must have DirectPass enabled for your organization to perform this type of data onboarding. If interested, contact your Account Manager.

ID filter added to Deployments page

The filter on the Deployments page has added the option to search by Deployment ID.
Deployments page details



January 23, 2026

Health Audience release note

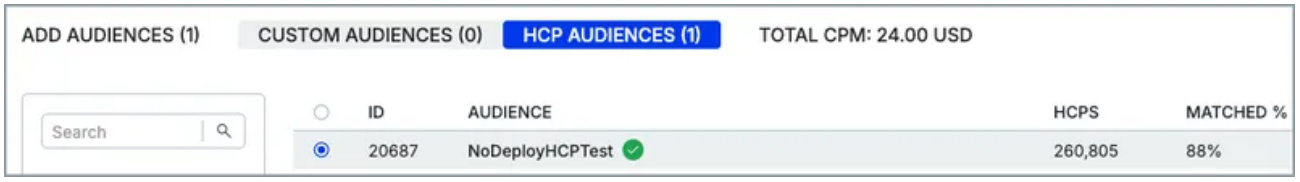
Assign an HCP audience to an ad group

What's an HCP audience?

A Healthcare Professional (HCP) audience is a target group consisting of healthcare professionals such as doctors, nurses, pharmacists, and dentists. When you build and save an HCP audience in the **Health Audience builder**, it's automatically available for assignment in the UI—as long as the audience and ad group are both associated with the same advertiser.

How to assign an HCP to an ad group

You can now assign an HCP audience to your ad group during the ad group setup. See **Add an ad group** for instructions.



Audience assignment screen in Ad Manager

6. # 5.4.1 DocumentSearchRequest Element Require

Onboarding Step by Step Process

These steps outline the process to onboard a customer to Surescripts Data Solutions, from account setup through data downloads.

ăndarăt

> **Note**: This is Early Adopter documentation. Requirements and content are subject to change.

Step 1: Verify Access to Data Package Builder in Workbench

- For instructions on how to set up access, see the **Setting Up Access to Workbench & Data Package Builder** instructions.

Step 2: Generate Keys

Step 2a: Generate RSA Key Pairs for Data Package Builder Authentication

To authenticate with the Data Package Builder API, you need to generate a 2048-bit RSA key pair and upload the public key through the Client Keys interface.

Option 1: Using Open SSL (Recommended - Works on Linux, macOS, Windows)

1. Generate a 2048-bit RSA private key.
2.

```
openssl genrsa -out private_key.pem 2048
```

2. Extract the public key.

```
openssl req -new -x509 -key privatekey.pem -out publickey509.pem -subj '/CN=data-package
```

3. Verify your keys pairs.

```
# View the private key
cat privatekey.pem
# View the public key
cat publickey509.pem
```

Option 2: Using PowerShell (Windows)

1. Generate RSA key pair.

```
$rsa = [System.Security.Cryptography.RSA]::Create(2048)
```

2. Export public key in X.509 SubjectPublicKeyInfo format.

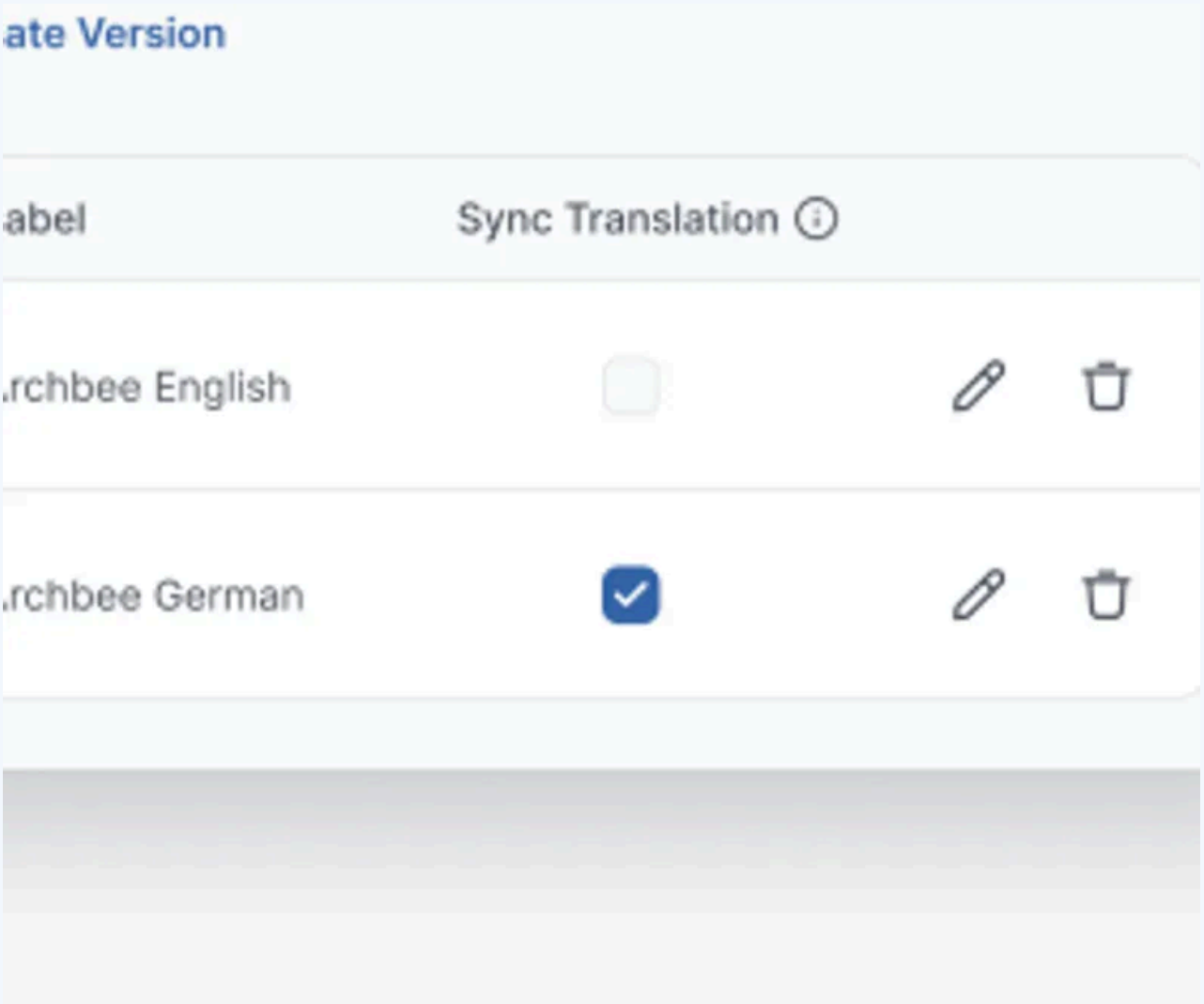
```
$publicKey = $rsa.ExportSubjectPublicKeyInfo()
$publicPem = "-----BEGIN PUBLIC KEY-----`n" +
             [Convert]::ToBase64String($publicKey, 'InsertLineBreaks') +
             "`n-----END PUBLIC KEY-----"
$publicPem | Out-File -FilePath "public_key.pem" -Encoding ASCII
```

3. Export private key.

```
$privateKey = $rsa.ExportRSAPrivateKey()
$privatePem = "-----BEGIN RSA PRIVATE KEY-----`n" +
              [Convert]::ToBase64String($privateKey, 'InsertLineBreaks') +
              "`n-----END RSA PRIVATE KEY-----"
$privatePem | Out-File -FilePath "private_key.pem" -Encoding ASCII

Write-Host "Keys generated successfully!"
Write-Host "Public key is in X.509 SubjectPublicKeyInfo format"
```

What You'll Get:



After generation of the key pair, you will have two files:

1. private_key.pem - Your private key
   - **CRITICAL**: Keep this secure
   - Store it in a secure location (HSM, key vault, encrypted storage)
   - This will be used to sign a compact JWT to request an acess token from the authorization server
2. public_key.pem - Your public key
   - This is what you upload to Data Package Builder
   - Safe to share (it's called "public" for a reason)
   - Example format:
   -----BEGIN PUBLIC KEY-----
   MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0Z3VS5JJcds3...
   -----END PUBLIC KEY-----

Step 2b: Upload your Public Key

1. Log in to **Data Package Builder**.
2. Navigate to **Client Keys**.
3. Paste the entire contents of **public_key.pem** (including the BEGIN/END lines).
4. Click **Create Client.**
    - **Note**: This will create the Client_ID that the customer will later use in **Build the Compact JWT Assertion** section.

⚠ **Important:** Once you have created your **Client ID**, notify your Surescripts point of contact. Surescripts must configure your data package before you can proceed to the next steps.

You will not be able to continue until this configuration is complete.

Security Best Practices:

✅ **DO**:

- Generate keys in a secure environment
- Store private keys in a key vault or HSM for production use
- Use strong file permissions on private key files (chmod 600 private_key.pem on Unix)
- Rotate keys periodically

❌ **DON'T**:

- Share or commit private keys to version control
- Email private keys
- Store private keys in plaintext in application code
- Reuse the same key pair across multiple clients/applications

Step 2c: Common errors

The below table includes information on common errors that can be encountered when creating key pairs.

Error	Troubleshooting Information
"Invalid certificate/public key format" error	• Ensure you're uploading the PUBLIC key, not the private key • Verify the PEM headers are included • Check for extra whitespace or line breaks
Key appears to upload but authentication fails	• Verify the corresponding private key is being used for signing requests • Ensure the key is either 2048 or 4096 (1024 is not allowed and larger key sizes will not fit in our 8 KB JWT input limit)

Step 3: Query Builder

The **Query Builder** in Workbench helps you create an **OData query URL** that defines exactly which prescribed medication data you want to retrieve from the Data Solutions API. You can build a query visually or edit it directly, depending on your preference.

Note: The Query Builder is optional; experienced users may submit **manual OData queries** directly.

OData is not the only type of request that is supported. For additional information on other uses for the Query Builder, see **Query Builder Overview** .

For additional information on OData, see **https://learn.microsoft.com/en-us/odata/overview**

Important: Before you begin:

- You must be able to **sign in to Workbench.**
- Your user must have access to **Data Package Builder** with the **Query Edit** role.
- You must be working with the **Prescribed Medication** product in Data Solutions.
- For instructions on this, see **Setting Up Access to Workbench & Data Package Builder**.

Step 3a: Run the Query Builder (OData)

1. Sign in to **Workbench** and open **Data Package Builder**.
2. Select **Query Builder**.
3. Choose the **Prescribed Medication** product.
4. Select the **columns** you want returned (these map to $select in OData).
5. Add **filters** to limit results.
    - A **PARTITION_DAY** filter is required.
6. Select **Generate from Selections** to automatically build the OData query
   (or manually edit the query if you're familiar with OData).
7. Copy the generated **OData query URL**.
8. Use the URL in your **Data Solutions API GET request** to retrieve Prescribed Medication data.

Note: For information on how to run the Query Builder for other use cases, see **Key Steps (Applies to All Use Cases)**

Step 4: Determining PARTITION_DAY Filter

To determine which PARTITION_DAY filter to request, see the **How to Use the Endpoints** page.

Step 5: Send REST Requests to Data Solutions API

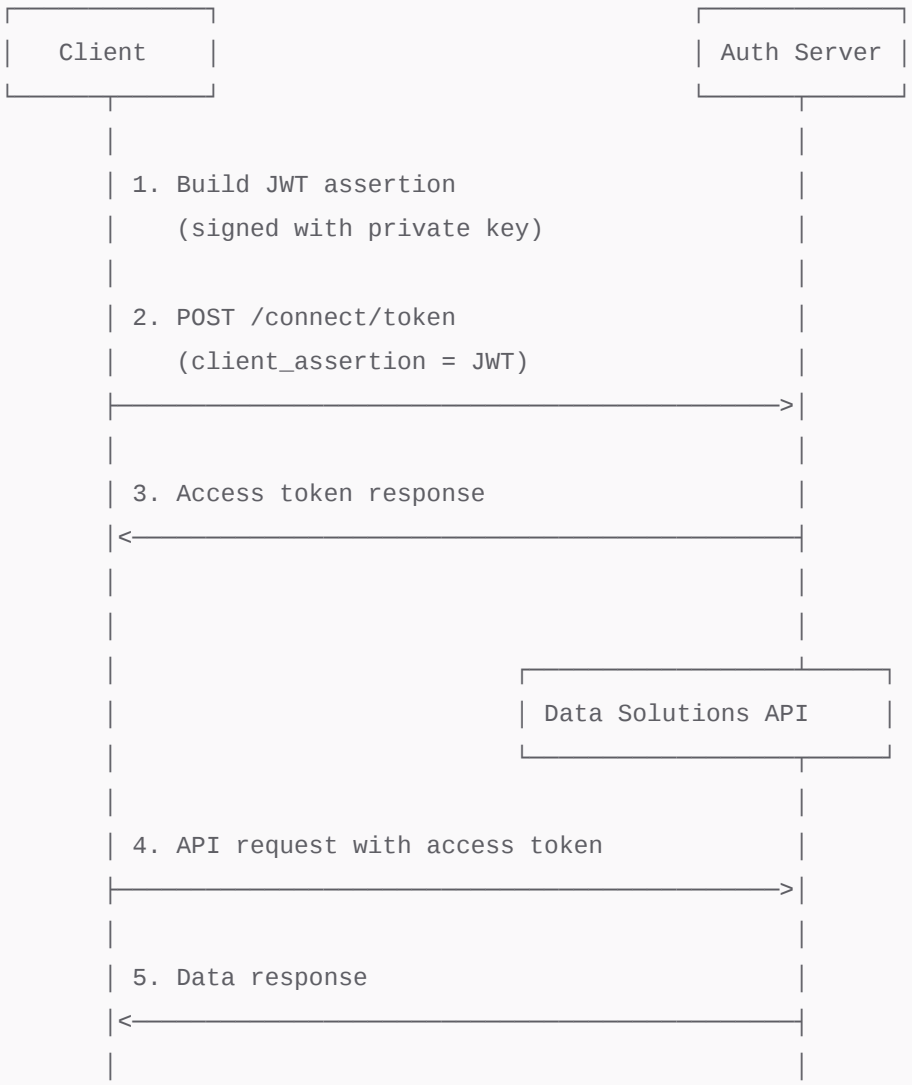
Authentication

Learn how to authenticate with the Data Solutions API using OAuth 2.0 Client Credentials Grant with JWT Bearer Assertion.

What You'll Do

The Data Solutions API uses a two-step authentication process:

1. **Build a JWT assertion** - Create and sign a JWT claim using your ClientID and private key
2. **Request an access token** - Exchange the JWT assertion for an OAuth access token at the auth-server's `/connect/token` endpoint
3. **Use the access token** - Include the access token in API requests



Before You Begin

Make sure you have:

1. **Client ID** - A unique identifier for your application.
2. **Private Key** - An RSA private key in **PKCS#8** PEM format.
3. **Public Key Registration** - Your public key must be registered with the auth server for your Client ID.

Private Key Format

Your private key must be in **PKCS#8 PEM format**:

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQC...
... base64 encoded key data ...
-----END PRIVATE KEY-----
```

Step 5a: Build the Compact JWT Assertion

Create a JWT with the following structure, signed with your private key using the RS384 algorithm (RSASSA-PKCS1-v1_5 with SHA-384). This JWT proves your identity to the auth server.

JWT Header

```
{
  "alg": "RS384",
  "typ": "JWT"
}
```

JWT Payload

```
{
  "iss": "your-client-id",                                    // Issuer: your client ID
  "sub": "your-client-id",                                    // Subject: your client ID
  "aud": "ENVIRONMENT_AUTH_URL/connect/token",                // Audience: token endpoint
  "jti": "550e8400-e29b-41d4-a716-446655440000",              // JWT ID: unique identifier
  "iat": 1706572800,                                          // Issued at: current Unix
  "nbf": 1706572800,                                          // Not before: current Unix
  "exp": 1706573100                                           // Expires: current timestamp
}
```

Important field notes:

- `iss` and `sub` must both be your Client ID
- `aud` must be the exact token endpoint URL (including `/connect/token`)
- `jti` must be unique for each request (use a UUID)
- `exp` should be 5 minutes from `iat` (recommended maximum)

Example: Building JWT in Python

```python
import jwt
import uuid
import time
from datetime import datetime, timedelta

# Your credentials
client_id = "your-client-id"
private_key = """-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQC...
-----END PRIVATE KEY-----"""
token_endpoint = "ENVIRONMENT_AUTH_URL/connect/token"

# Build the JWT payload
now = int(time.time())
payload = {
    "iss": client_id,
    "sub": client_id,
    "aud": token_endpoint,
    "jti": str(uuid.uuid4()),
    "iat": now,
    "nbf": now,
    "exp": now + 300  # 5 minutes from now
}

# Sign the JWT with RS384
client_assertion = jwt.encode(
    payload,
    private_key,
    algorithm="RS384"
)

print(f"JWT Assertion: {client_assertion}")
```

Example: Building JWT in C#

```
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Security.Cryptography;
using Microsoft.IdentityModel.Tokens;

// Your credentials
var clientId = "your-client-id";
var privateKeyPem = @"-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQC...
-----END PRIVATE KEY-----";
var tokenEndpoint = "ENVIRONMENT_AUTH_URL/connect/token";

// Load the private key
using var rsa = RSA.Create();
rsa.ImportFromPem(privateKeyPem);
var signingCredentials = new SigningCredentials(
    new RsaSecurityKey(rsa),
    SecurityAlgorithms.RsaSha384
);

// Build the JWT
var now = DateTime.UtcNow;
var handler = new JwtSecurityTokenHandler();
var token = handler.CreateJwtSecurityToken(
    issuer: clientId,
    audience: tokenEndpoint,
    subject: new ClaimsIdentity(new[] { new Claim("sub", clientId) }),
    notBefore: now,
    expires: now.AddMinutes(5),
    issuedAt: now,
    signingCredentials: signingCredentials
);

// Add the jti claim
token.Payload["jti"] = Guid.NewGuid().ToString();

var clientAssertion = handler.WriteToken(token);
Console.WriteLine($"JWT Assertion: {clientAssertion}");
```

Example: Building JWT in JavaScript/Node.js

```
const jwt = require('jsonwebtoken');
const { v4: uuidv4 } = require('uuid');
const fs = require('fs');

// Your credentials
const clientId = 'your-client-id';
const privateKey = fs.readFileSync('private_key.pem', 'utf8');
const tokenEndpoint = 'ENVIRONMENT_AUTH_URL/connect/token';

// Build the JWT payload
const now = Math.floor(Date.now() / 1000);
const payload = {
  iss: clientId,
  sub: clientId,
  aud: tokenEndpoint,
  jti: uuidv4(),
  iat: now,
  nbf: now,
  exp: now + 300 // 5 minutes
};

// Sign the JWT with RS384
const clientAssertion = jwt.sign(payload, privateKey, {
  algorithm: 'RS384'
});

console.log(`JWT Assertion: ${clientAssertion}`);
```

Step 5b: Request Access Token

Send a POST request to the auth server's `/connect/token` endpoint with the JWT assertion.

Request Format

```
POST /connect/token HTTP/1.1
Host: ENVIRONMENT_AUTH_URL
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
&client_id=your-client-id
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=<your-signed-jwt>
&scope=DsaApi
```

Parameters:

- `grant_type` - Must be `client_credentials`
- `client_id` - Your client identifier
- `client_assertion_type` - Must be `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`
- `client_assertion` - The signed JWT from Step 1
- `scope` - Must be `DsaApi` to access the Data Solutions API

Example: Request Token in Python

```python
import requests

# Build the token request
token_url = "ENVIRONMENT_AUTH_URL/connect/token"
data = {
    "grant_type": "client_credentials",
    "client_id": client_id,
    "client_assertion_type": "urn:ietf:params:oauth:client-assertion-type:jwt-bearer",
    "client_assertion": client_assertion,
    "scope": "DsaApi"
}

# Request the access token
response = requests.post(token_url, data=data)
response.raise_for_status()

token_response = response.json()
access_token = token_response["access_token"]
expires_in = token_response["expires_in"]

print(f"Access Token: {access_token}")
print(f"Expires in: {expires_in} seconds")
```

Example: Request Token in C#

```
using System.Net.Http;

var httpClient = new HttpClient();
var tokenUrl = "ENVIRONMENT_AUTH_URL/connect/token";

var requestData = new Dictionary<string, string>
{
    { "grant_type", "client_credentials" },
    { "client_id", clientId },
    { "client_assertion_type", "urn:ietf:params:oauth:client-assertion-type:jwt-bearer" }
    { "client_assertion", clientAssertion },
    { "scope", "DsaApi" }
};

var response = await httpClient.PostAsync(tokenUrl, new FormUrlEncodedContent(requestDat
response.EnsureSuccessStatusCode();

var tokenResponse = await response.Content.ReadFromJsonAsync<TokenResponse>();
var accessToken = tokenResponse.AccessToken;
var expiresIn = tokenResponse.ExpiresIn;

Console.WriteLine($"Access Token: {accessToken}");
Console.WriteLine($"Expires in: {expiresIn} seconds");

// Token response model
public class TokenResponse
{
    [JsonPropertyName("access_token")]
    public string AccessToken { get; set; }

    [JsonPropertyName("expires_in")]
    public int ExpiresIn { get; set; }

    [JsonPropertyName("token_type")]
    public string TokenType { get; set; }
}
```

Example: Request Token in JavaScript/Node.js

```node
const axios = require('axios');
const qs = require('querystring');

const tokenUrl = 'ENVIRONMENT_AUTH_URL/connect/token';

const requestData = {
  grant_type: 'client_credentials',
  client_id: clientId,
  client_assertion_type: 'urn:ietf:params:oauth:client-assertion-type:jwt-bearer',
  client_assertion: clientAssertion,
  scope: 'DsaApi'
};

axios.post(tokenUrl, qs.stringify(requestData), {
  headers: { 'Content-Type': 'application/x-www-form-urlencoded' }
})
.then(response => {
  const accessToken = response.data.access_token;
  const expiresIn = response.data.expires_in;

  console.log(`Access Token: ${accessToken}`);
  console.log(`Expires in: ${expiresIn} seconds`);
})
.catch(error => {
  console.error('Token request failed:', error.response?.data || error.message);
});
```

Successful Response

```node
{
  "access_token": "eyJhbGciOiJSUzM4NCIsImtpZCI6IjEyMzQ1Njc4OTAiLCJ0eXAiOiJKV1QifQ...",
  "expires_in": 3600,
  "token_type": "Bearer",
  "scope": "DsaApi"
}
```

Error Response

```
{
  "error": "invalid_client",
  "error_description": "Invalid client or client credentials"
}
```

Common error codes:

- `invalid_client` - Client ID not found or public key mismatch
- `invalid_grant` - JWT assertion validation failed
- `invalid_scope` - Requested scope not available

Step 5c: Use Access Token in Data Solutions API Requests

Include the access token in the `Authorization` header of your API requests.

Request Format

```
GET /api/v0/odata/PrescribedMedicationData?$filter=PARTITION_DAY eq '2025-01-30' HTTP/1.1
Host: ENVIRONMENT_DATA_SOLUTIONS_URL
Authorization: Bearer <your-access-token>
```

Example: API Request in Python

```python
import requests

api_url = "ENVIRONMENT_DATA_SOLUTIONS_URL/api/v0/odata/PrescribedMedicationData"
headers = {
    "Authorization": f"Bearer {access_token}"
}
params = {
    "$filter": "PARTITION_DAY eq '2025-01-30'",
    "$top": 10
}

response = requests.get(api_url, headers=headers, params=params)
response.raise_for_status()

data = response.json()
print(f"Retrieved {len(data['value'])} records")
```

Example: API Request in C#

```
var apiUrl = "ENVIRONMENT_DATA_SOLUTIONS_URL/api/v0/odata/PrescribedMedicationData";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization =
    new AuthenticationHeaderValue("Bearer", accessToken);

var queryParams = new Dictionary<string, string>
{
    { "$filter", "PARTITION_DAY eq '2025-01-30'" },
    { "$top", "10" }
};

var queryString = string.Join("&", queryParams.Select(kvp =>
    $"{Uri.EscapeDataString(kvp.Key)}={Uri.EscapeDataString(kvp.Value)}"));
var requestUrl = $"{apiUrl}?{queryString}";

var response = await httpClient.GetAsync(requestUrl);
response.EnsureSuccessStatusCode();

var data = await response.Content.ReadFromJsonAsync<ODataResponse>();
Console.WriteLine($"Retrieved {data.Value.Count} records");
```

Example: API Request in JavaScript/Node.js

```
const axios = require('axios');

const apiUrl = 'ENVIRONMENT_DATA_SOLUTIONS_URL/api/v0/odata/PrescribedMedicationData';
const headers = {
  'Authorization': `Bearer ${accessToken}`
};
const params = {
  '$filter': "PARTITION_DAY eq '2025-01-30'",
  '$top': 10
};

axios.get(apiUrl, { headers, params })
  .then(response => {
    console.log(`Retrieved ${response.data.value.length} records`);
  })
  .catch(error => {
    console.error('API request failed:', error.response?.data || error.message);
  });
```

Step 5d: Handle Token Expiration

Access tokens expire after the time specified in the `expires_in` field (typically 3600 seconds / 1 hour).

Best practices:

1. Store the token and its expiration time
2. Check if the token is expired before each request
3. Request a new token when the current one expires or is about to expire
4. Add a buffer (e.g., refresh 5 minutes before actual expiration)

Example: Token Management in Python

```python
from datetime import datetime, timedelta

class TokenManager:
    def __init__(self, client_id, private_key, token_endpoint):
        self.client_id = client_id
        self.private_key = private_key
        self.token_endpoint = token_endpoint
        self.access_token = None
        self.expires_at = None

    def get_access_token(self):
        # Return cached token if still valid
        if self.access_token and datetime.utcnow() < self.expires_at:
            return self.access_token

        # Request new token
        client_assertion = self._build_jwt()
        response = requests.post(self.token_endpoint, data={
            "grant_type": "client_credentials",
            "client_id": self.client_id,
            "client_assertion_type": "urn:ietf:params:oauth:client-assertion-type:jwt-bea
            "client_assertion": client_assertion,
            "scope": "DsaApi"
        })
        response.raise_for_status()

        token_response = response.json()
        self.access_token = token_response["access_token"]

        # Set expiration with 5-minute buffer
        expires_in = token_response["expires_in"]
        self.expires_at = datetime.utcnow() + timedelta(seconds=expires_in - 300)

        return self.access_token

    def _build_jwt(self):
        # JWT building logic from Step 1
        now = int(time.time())
        payload = {
            "iss": self.client_id,
            "sub": self.client_id,
            "aud": self.token_endpoint,
            "jti": str(uuid.uuid4()),
```

```
            "iat": now,
            "nbf": now,
            "exp": now + 300
        }
        return jwt.encode(payload, self.private_key, algorithm="RS384")

# Usage
token_manager = TokenManager(client_id, private_key, token_endpoint)
access_token = token_manager.get_access_token()
```

Optional: Use DPoP for Enhanced Security

The API supports DPoP (Demonstration of Proof-of-Possession) for enhanced security. DPoP binds access tokens to a specific key pair, preventing token theft.

DPoP Token Request

Include a DPoP proof JWT in the `DPoP` header:

```
POST /connect/token HTTP/1.1
Host: auth-server.surescripts.net
Content-Type: application/x-www-form-urlencoded
DPoP: <dpop-proof-jwt>

grant_type=client_credentials
&client_id=your-client-id
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=<your-signed-jwt>
&scope=DsaApi
&dpop_jkt=<jwk-thumbprint>
```

DPoP Proof Structure

Header:

```
{
  "typ": "dpop+jwt",
  "alg": "RS256",
  "jwk": {
    "kty": "RSA",
    "n": "<base64url-modulus>",
    "e": "<base64url-exponent>"
  }
}
```

Payload:

```
{
  "htu": "ENVIRONMENT_AUTH_URL/connect/token",
  "htm": "POST",
  "iat": 1706572800,
  "jti": "unique-id-here"
}
```

DPoP API Request

When using a DPoP-bound token, include the DPoP proof in API requests:

```
GET /api/v0/odata/PrescribedMedicationData HTTP/1.1
Host: ENVIRONMENT_DATA_SOLUTIONS_URL
Authorization: DPoP <access-token>
DPoP: <dpop-proof-jwt>
```

The DPoP proof for resource requests must include the `ath` claim (access token hash):

```
{
  "htu": "ENVIRONMENT_DATA_SOLUTIONS_URL/api/v0/odata/PrescribedMedicationData",
  "htm": "GET",
  "iat": 1706572800,
  "jti": "unique-id-here",
  "ath": "<base64url-sha256-of-access-token>"
}
```

For detailed DPoP implementation, see the **Postman Collection README**.

Troubleshooting

"invalid_client" Error

Cause: Client ID not found or public key mismatch

Solutions:

1. Verify your Client ID is correct
2. Ensure your public key is registered with the auth server
3. Confirm the private key matches the registered public key

"invalid_grant" Error

Cause: JWT assertion validation failed

Solutions:

1. Check JWT expiration (`exp` claim) - ensure it's in the future
2. Verify `aud` claim matches the exact token endpoint URL
3. Ensure `iss` and `sub` both contain your Client ID
4. Confirm JWT is signed with RS384 algorithm
5. Check that JWT `jti` is unique

401 Unauthorized on API Requests

Cause: Invalid or expired access token

Solutions:

1. Verify the token hasn't expired
2. Check that you're using `Bearer` token type
3. Ensure the token was requested with `scope=DsaApi`
4. Confirm the `Authorization` header format: `Bearer <token>`

403 Forbidden on API Requests

Cause: Valid token but insufficient permissions

Solutions:

1. Verify your client has purchased the product/package
2. Check if the columns you're selecting are in your package
3. Ensure you're including a valid PARTITION_DAY filter
4. Review package filter restrictions

Token Request Timeout

Cause: Network connectivity or server issues

Solutions:

1. Check network connectivity to the auth server
2. Verify the auth server URL is correct
3. Check firewall rules allow outbound HTTPS

Environment URLs

Environment	Auth Server URL	Data Solutions API URL
Production	`https://auth-server.surescripts.net`	`https://data-solutions-api.surescripts.net`

Security Best Practices

1. **Protect Private Keys**
    - Never commit private keys to source control
    - Store keys securely (e.g., environment variables, key vaults)
    - Use different keys for different environments
2. **Token Storage**
    - Store tokens securely in memory or encrypted storage
    - Never log or expose tokens in error messages
    - Clear tokens when they expire
3. **JWT Claims**
    - Always use unique `jti` values (UUIDs)
    - Keep expiration times short (5 minutes recommended)
    - Validate token expiration before use
4. **HTTPS Only**
    - Always use HTTPS for token requests and API calls
    - Verify SSL/TLS certificates
5. **Error Handling**
    - Implement retry logic for transient failures
    - Log authentication failures for monitoring
    - Don't expose sensitive details in client-facing errors

Step 6: Download Data Files (Optional Formats)

The Data Solutions API supports optional file download formats that may be enabled for your data package. Available formats depend on your package configuration.

Supported Formats

- **CSV** – Human-readable format for simple downloads and manual review
- **Parquet** – Binary, columnar format optimized for analytics and large-scale processing

> **Note:** File format availability is controlled at the package level by Surescripts. If a format is not enabled for your package, it is not available for download.

Before You Begin

Confirm that:

- Your Client ID is configured with a package that supports the desired file format
- You have completed authentication in **Step 5**
- You have a valid **PARTITION_DAY** value

File format selection does **not** affect query construction or authentication.

Download CSV Data (For Enabled CSV Packages Only)

Send a request to the following endpoint to download CSV data using a signed URL. The signed URL is active for 5 minutes and is only available is you are enabled for CSV downloads.

Prescribed Medication Data:

```
/api/v0/customer/product/PrescribedMedicationData/csv/signedurls/{PARTITION_DAY}
```

Download Parquet Data (For Enabled Parquet Packages Only)

Send a request to the following endpoint to download Parquet data using a signed URL. The signed URL is active for 5 minutes and is only available is you are enabled for Parquet downloads.

Prescribed Medication Data:

```
/api/v0/customer/product/PrescribedMedicationData/parquet/signedurls/{PARTITION_DAY}
```

7. test copy paste

7.1. 5.4.1 DocumentRetrieveRequest Element Requirements

API Information

oknjknjknj

The DocumentRetrieveRequest is a GET message that will use the following web service API endpoint information to retrieve a document through Surescripts.

Operation	API Endpoint Description	Description
Document Retrieve	GET [base]/Binary?documentID=&repository Id>&homeCommunityId=	Returns the document in binary form.

> **Note**: As a responder you will be responsible for defining the endpoint information and URL for others to retrieve documents from your system. That information is returned in the DocumentSearchResponse from your system.

Content Types and Encodings

According to Section 3.1.0.1.9 of the FHIR® specification Content Types and encodings,

“The formal MIME-type for FHIR® resources is application/fhir+json or application/fhir+xml. The correct mime type SHALL be used by clients and servers.” Later in that same section, “UTF-8 encoding SHALL be used for the mime type application/fhir. This MAY be specified as a MIME type parameter to the application/fhir mime type, but is not required” because

FHIR® uses UTF-8 for all request and response bodies. Since the HTTP specification (section 3.7.1) defines a default character encoding of ISO-8859-1, requests and responses SHALL explicitly set the character encoding to UTF-8 using the charset parameter of the MIME-type in the Content-Type header. Requests MAY also specify this charset parameter in the Accept header and/or use the Accept-Charset header."

Because of these requirements, requests/responses will need to include a Content-Type in the HTTP header. Below is an example:

- Content-Type: application/fhir+json; charset=utf-8

> **Note**: charset=utf-8 is not required by Surescripts, but is supported if sent.

Accept Header content v. the_format Request Parameter

The FHIR R4 specifications state that documents should be returned based on how they were requested which could be either via Bundles with Base64 encoded contents within a bundle or as streamed content. The Accept request-header field within a document retrieve request can be used to specify the media types that are acceptable for each response.

- If "application/fhir+xml" or "application/fhir+json" is specified in a request header's Accept field, the document will be returned via FHIR bundles.
    - Documents (ex: CCDAs, PLQs, binary files) that are returned to document requesters via bundles within the FHIR responses need to be Base64 encoded
- If other, ex: "application/pdf" (or other binary types), the requested document will be streamed directly to the requester

Alternatively, the FHIR® RLE product also supports the use of the "_format" parameter as described in Section 3.1.0.1.11 of the FHIR® specification as a way for document requestors to specify their preference to receive requested documents in a format other than what is included in the HTTP header (for situations where a requesting application can control the contents of their requests, but not the HTTP header due to the use of 3rd party communication software or other limitations).

Request example with "_format" parameter:

https://rle.surescripts.net/fhir/R4/Binary?documentId=CCDA123&repositoryId=urn:

oid:44.5555.66=urn:oid:77.8888.99&

_format=application/fhir+json

Be sure to note that use of the “_format” parameter in the request overrides any media request types specified in a request’s message header Accept field.

5.4.10 DocumentRetrieveRequest Example

The following URL and HTTP header example would return a bundled response in the JSON format.

https://rle.surescripts.net/fhir/Binary?documentId=2.16.840.1.113883.3.2054.2.1.128118651&repositoryId=2.16.840.1.113883.3.2054.2.1&homeCommunityId=2.16.840.1.113883.3.2054.2.1

```
Accept: application/fhir+json
Accept-Encoding: gzip,deflate
Connection: keep-alive
Content-Type: application/fhir+json
Originator-Assigning-Authority: 2.11.222.3.444444.5.666.7.2
Originator-Home-Community-Id: urn:oid:aFRVAy
Originator-Message-Id: 3bb7cae9-e835-4844-a3ae-1c41abcea564
Originator-Organization-Name: ABC Medical Software
Originator-Patient-Identifier: 1002318246
Originator-User-Id: SMJO1234579
Originator-User-Name: ABC User
Originator-User-Role-Code: 106289002
Purpose-Of-Use-Code: TREATMENT
User-Agent: Apache-HttpClient/4.5.13 (Java/11.0.5)
X-SSL-Client-Cert-Issuer: ABC Cert Issuer
X-SSL-Client-Cert-Subject: ABC Cert Subject
```

5.4.11 DocumentRetrieveResponse Element Requirements

Documents will be returned based on how they were requested (see DocumentRetrieveRequest) which could either be via Bundles with Base64 encoded contents within a bundle, or as content streamed directly to the requestor in the format (without encoding) that was specified by the contentType of the requested document.

The table below outlines the requirements for the Bundle resource used in the DocumentRetrieveResponse message.

Note: This guide includes data elements only where Surescripts has specific requirements or further explains the field usage. Refer to the Bundle FHIR resource for a complete list of fields.

Give this API endpoint a title...

Get a cake by its ID

GET https://api.cakes.com

Request Response

BODY PARAMETERS

resource type String required

Identifies this as a Bundle resource.

id String optional

Logical identifier for the response bundle.

type String required

Indicates the purpose of this bundle – how it is intended to be used.
Fixed value: searchset

entry String required

Search results returned by the DocumentSearch operation.

resource String optional

A resource that represents the data of a single raw artifact as digital content accessible in its native format. A Binary resource can contain any content, whether text, image, pdf, zip archive, etc.
Comments
Typically, Binary resources are used for handling content such as:

- CDA Documents (i.e. with XDS)
- PDF Documents
- Images (the Media resource is preferred for handling images, but not possible when the content is already binary - e.g. XDS).

resourceType String optional

Fixed value: Binary

id String optional

Logical identifier for the Binary resource.

contentType String optional

MIME type of the document (e.g., image/jpeg, application/pdf).

data String optional

Base64-encoded document content.


```
{
  "name": "Cake's name",
}
```


5.4.12 DocumentRetrieveResponse Message Example

```
{
  "resourceType": "Bundle",
  "id": "f98511eb-a5a2-4ecf-8ee9-c6d35bb3bb99",
  "type": "searchset",
  "entry": [
    {
      "resource": {
        "resourceType": "Binary",
        "id": "16dade3d-e45b-4be4-889b-c0ca65bc9367",
        "contentType": "image/jpeg",
        "data": "iVBORw0KGgoAAAANSUhEUgAAAmQAAAC0CAYAAAA6sDC6…(Base64 content truncated
      }
    }
  ]
}
```

8. Dairy Comp page(VAS) with duplicated content (cloned with children) (cloned with children)

Expand the sections below to learn about new features, enhancements, and bug fixes in the latest DairyComp 305 release!

DairyComp - Version 26.3

Release Date: April 1, 2026

DairyComp version 26.3 contains various bug fixes, customer-requested enhancements, and improvements.

Expand the topics below to learn more about the updates in this release:

Integrations

Zoetis:

Added support for additional Zoetis genomic traits. These traits are received through the VAS Developer API and can be mapped in DairyComp using the PLATFORM command for reporting. See **steps for storing PULSE genomic items in DairyComp** for instructions.

Item Names:

- **GZRFI:** Zoetis Residual Feed Intake
- **GZMI:** Zoetis Milk Methane Intensity
- **GZFR:** Zoetis Fertility Heat Resilience
- **GZMR:** Zoetis Milk Heat Resilience
- **GDWPH:** Zoetis Dairy Wellness Profit with Heat Resilience
- **GZRFI:** Zoetis Residual Feed Intake
- **GZMI:** Zoetis Milk Methane Intensity
- **GZFR:** Zoetis Fertility Heat Resilience
- **GZMR:** Zoetis Milk Heat Resilience
- **GDWPH:** Zoetis Dairy Wellness Profit with Heat Resilience

CowCare

- We have fixed an issue that prevented the CowCare Purchase Report from populating correctly. The report now works as expected.

Base Program

- VETLIST right column IDs are now clickable when the `\P` switch is used.
- The command `EVENTS\3 by %190.PSIRC.7` now displays semen type in the report.
- You can now add your company logo to INVOICE reports. See **Automate Billing Reports using INVOICE** to learn more.
- Indicators were added to the **Connection Status** tab to quickly show if your connections are healthy without having to go to the Connection Status dashboard. Integration cards that are hidden do not affect the status:

- In-app help links were added for these functions:
  - **EGRAPH**
  - **GRAPH**
  - **ALTER5 : VetList Options**
- In the **User Manager**, COWVAL was added as a user-definable command.
- The `Events\12A` command was added to the Erase Events function in the **User Manager**. Now, if Erase Events is checked for any user level, `Events\12A` is disabled.
- German translations for DairyComp were updated.
- Beef calves are now excluded from `ECON\I` and `ECON\IR` reports.
- December 2025 NAAB proof updates were applied for the US, UK, Germany, and Canada.
- Resolved an issue that caused information to be truncated when hovering over the `PARLOR\W` report.
- Resolved an issue that occurred when entering beef heifers using the FRESH event.
- Resolved an issue with the default report view so the report view remains visible.
- Resolved an issue where sire cleanup changed the semen type incorrectly for German users.

New FAQ: Visualize Test Day Performance Trends with PLOT

We added a new help topic, **Visualize Test Day Performance Trends with PLOT**, that shows you how to graph milk and component trends in DairyComp. This topic explains how to plot individual or group test day data, compare groups, and adjust settings for detailed performance insights.

New FAQ: Automate Billing Reports using INVOICE

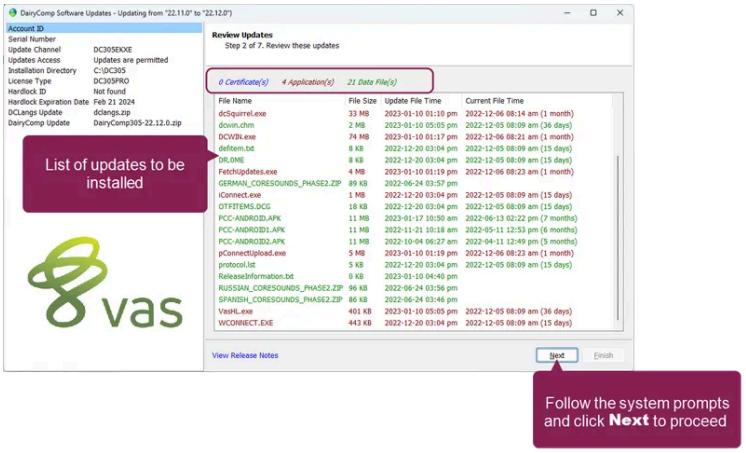
We added a new help topic, **Automate Billing Reports using INVOICE**, that shows you how to generate customized invoices in DairyComp. It covers creating and managing invoice definition files, running the INVOICE command, and using options for filtering.

Need help updating DC305? Click here to see how it's done!

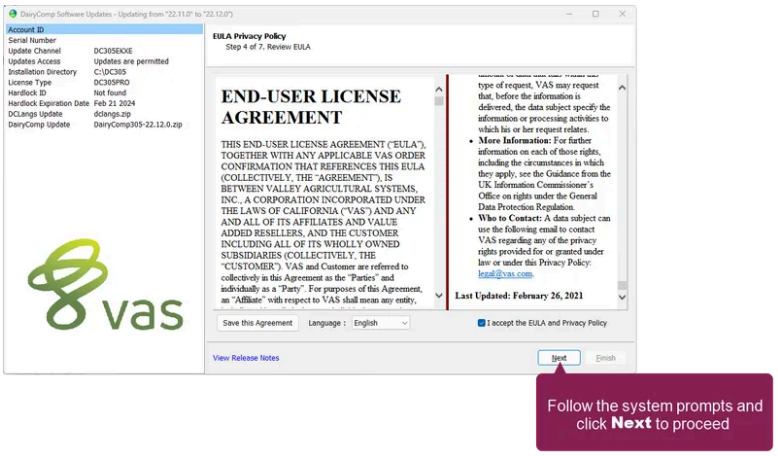
I've installed updates on DC305 before. Remind me how!

1. Open DC305.
2. Select Help > Check for Updates from the menu bar.
   This launches the Web Updater. After checking for updates, it shows the

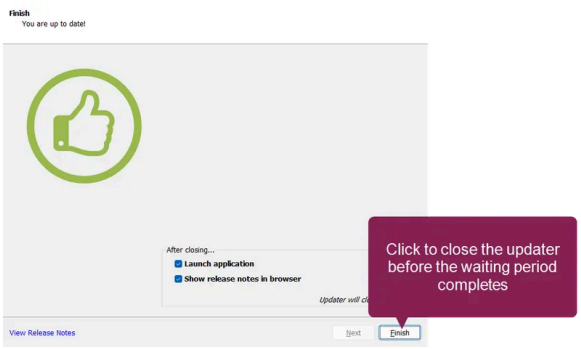
available updates by category and lists all of the updates between your current version and the latest version.



3. Follow the system prompts to navigate through the remaining steps of the update process. This may involve accepting VAS legal agreements, which is typically only required once per year or when the terms of a policy are updated.



4. When the updater finishes, it notifies you that your system is up-to-date. After 15 seconds, the updater automatically closes and launches DC305. Or, you can click **Finish** to immediately launch DC305.



I've never updated DC305. Show me how!

If you're not familiar with the DC305 Web Updater, you can watch the following video tutorial for instructions on how to check your current DC305 version and then check for updates in the latest version. Note that the UI may differ between the video and your current DC305 version.

Earlier Releases

DairyComp - Version 25.12

Release Date: January 6, 2026

DairyComp version 25.12 contains various bug fixes, customer-requested enhancements, and improvements.

Expand the topics below to learn more about the updates in this release:

DairyComp 305 - Version 25.11

Release Date: December 2, 2025

DC305 version 25.11 contains various bug fixes, customer-requested enhancements, and integration improvements for our integration partners, as well as many other improvements.

Expand the topics below to learn more about the updates in this release:

Base Program

- DC305 can now capture event entry dates and times. Use the following new items to track event entry (version 10 cowfiles only):
    - Item type 175: A calculated item that extracts the date the event was entered into DC305. This differs from the actual event date, because events are often recorded on a date that is not the same as when they occurred for the animal.
    - Item type 176: A calculated item that extracts the time the event was entered into DC305. This is the time the event was entered into DC305, not the time the event occurred for the animal.
    - Event entry date and time can be see in the event editor for each event on the CowCard. This field is read only. Events that existed prior to this release show the event date and `00:00` for the time.

- DC305 now supports additional item details:
    - Item type 124 now has a new column that can be used to show total lactation production components (i.e., the sum of fat and protein). Choose **Fat+Ptn** for the column to see total lactation production components:

    - Item type 125 now has additional Operand 2 options that can be used to show total lactation as a days in milk (DIM) value for fat, protein, and the sum of fat and protein.
      For example, `PRT30` with `Operand 1:30` is the total lactation protein as of DIM 30:

    - Item type 126 now has additional Operand 2 options that can be used to show total lactation at the time of a specific date item for fat, protein, and the sum of fat and protein.
      For example, `POCOM` with `Item Name: PODAT` and `Operand 2: Fat+Ptn` returns the sum of fat and protein at the time of the PODAT

(preg/open date):

- Resolved an issue that caused the DC Client task table to be overwritten by the DC Server.
- Updated the 646 stud code to show as sexed semen.
- Russian language vet codes are now supported in the **ALTER\5** function.
- **Integrated help** has been added to the **SERVER** button when there are **HELD commands**.
- The `SAVE\2` option in **SETUP** now allows you to set the number of backups to keep.
- When the `\GG` switch is used, item type 54 in the **EVENTS** function now displays the item data for the corresponding event.
- The PLOT function now sets the minimum y-axis value for Somatic Cell Count (SCC) charts to 50 (was 0). This improves the visual contrast and readability of stacked multi-bar charts involving SCC.

- Item type 130 in the **EVENTS** function now displays correctly.
- When the `\GG` switch is used, item type 54 in the **EVENTS** function now displays the item data for the corresponding event.
- A default printer is no longer required. The following message now displays: Printing is disabled because no default printer has been found. To stop seeing this message, add a default printer or disable printing:
  ADD A PRINTER: In Windows Settings, add or select a default printer. Restart DairyComp and clear the message. Enter SETUP in the command line. In the Printer tab, check the box next to Print Enabled. Click OK to save your changes.
  DISABLE PRINTING: Enter SETUP in the command line. In the Printer tab, check the box next to Disable All Printing. Click OK to save your changes.
- Resolved an issue with the event translation file when it was automatically created.
- Resolved an issue with a corrupted DCGUIDE file.
- Resolved an issue causing license errors for Russian customers.

Integrations

- Jantec: Updated handling of Jantec milkings when a milk shift crosses midnight. **Contact support** for assistance if your setup has a milk shift that crosses midnight.

Pocket CowCard

PCC Android - Version v2509081615-(2.0.33.7-rc): When you send or receive data from DC305, PCC automatically sends the device name, PCC version, and Android OS version so you can better log updates and troubleshoot any device issues. Example log issues:

- `S 01:19 PCC 9ca667a077cd2971 - Gary's S21+ - v2507011612-(2.0.30.237-rc) A15 - V10 refresh started`
- `S 01:19 PCC 9ca667a077cd2971 - Gary's S21+ - v2507011612-(2.0.30.237-rc) A15 - V10 refresh completed`

New Help Topic: ALTER\3: Command Abbreviations

We added a new help topic, **ALTER\3: Command Abbreviations**, that describes how to manage command abbreviations in DC305. The topic covers how to create, edit, delete, and test command abbreviations, as well as options such as printing or exporting your abbreviation list and saving or reading commands from a file.

New FAQ: Visualize Item Distribution with GRAPH

We added a new help topic, **Visualize Item Distribution with GRAPH**, that shows you how to analyze numeric animal items in DC305 using graphical tools. The topic explains how to create bar graphs for frequency distributions and scatter plots for correlations between items. This helps you quickly visualize herd data and identify trends or relationships.

New FAQ: Visualize Event Distribution with EGRAPH

We added a new help topic, **Visualize Event Distribution with EGRAPH**, that provides guidance on graphing herd events in DC305. The topic covers creating bar graphs and scatter plots, customizing parameters, and drilling down into event details to help monitor trends and validate protocols.

New FAQ: Visualize Historical Measurements with EPLOT

We added a new help topic, **Visualize Historical Measurements with EPLOT**, which explains how to graph trends for body condition score, weight, and height in DC305. The topic covers plotting individual or group measurements over time and customizing settings for accurate analysis.

DairyComp 305 - Version 25.10

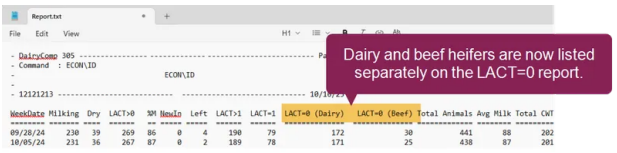
Release Date: October 29, 2025

DC305 version 25.10 contains various bug fixes, customer-requested enhancements, and integration improvements for our integration partners, as well as many other improvements.

Expand the topics below to learn more about the updates in this release:

Base Program

- Added support for the Canadian sire file updates in the upcoming December 2025 proofs.
- iTrans now ignores the Event Gap setting when sending events.
- DC305 now includes in-app links to the DC305 Online Help pages for **HELD Commands** and **Events**.
- The Guide report for "What is the testday estimated bulk tank SCC?" now works as expected.
- The DC Client update banner now directs users to the DC Server when a new version update is available to download.
- Support has been added for multiple DNB vet codes.
- The Y-axis label for `EGRAPH \S` has been updated.
- Added a second prompt for "Live Female Percentage for Sexed Dairy Semen" to `ECON\IR`, as well as updated `ECON\I` report text.
- A GETCOW issue no longer changes the date format on items.
- Improved PowerPoint readability by updating the slide title font, so numbers in the title are no longer subscript.
- The `ECON\ID` report LACT=0 inventory now displays dairy heifers and beef separately to follow CBRD standards. In the report, they are listed as LACT=0 (Dairy) and LACT=0 (Beef):



- iReports no longer include unnecessary blank slides.
- The ABS Global stud code 929 is now accurately labeled as semen type Q.
- The **Assist** feature now works correctly when Russian is the chosen language.

Integrations

- Genomics:
  - Added support for genomics partners that add breed indexes to the genomics file. DairyComp will continue to calculate the breed index if breed index values are not included in the imported genomics file.
  - Added Holstein Milking Speed to the genomics import file via DNG and PULSE. You can use the command `IMPORTF\G` to map or create an item to populate Holstein Milking Speed PTA:
-
- Germany: Fixed an issue where `SEND\D` was missing the calving event if lactations started as a result of an abort.
- Lely:
  - Improved DCSquirrel, so it now ignores an ANID assignment if the animal ID is null.
  - DC305 now checks whether an ANID is an eligible Item Type #32.
- Amelicor: Increased the ID length to 6 digits for the Amelicor DRPC.
- Parlors: 9-digit animal IDs are now supported in the milking report's Wrong Pens section.
- Lactanet DairyTrace: When you are reporting a MOVEIN record, TRACE reporting in DairyComp will now warn you if you have a blank EDAT. The record won't be sent until a date is entered.

DairyComp 305 - Version 25.9

Release Date: October 7, 2025

DC305 version 25.9 contains various bug fixes, customer-requested enhancements, and integration improvements for our integration partners, as well as many other improvements.

Expand the topics below to learn more about the updates in this release:

New Feature: Assist

DC305 now includes Assist, a smarter way to use the command line. As you type, Assist suggests matching items, events, reports and GUIDE questions, making it easier to build commands quickly and accurately. See **Assist** to learn more!

New Feature: Connection Status

Check your integrations at a glance with the new Connection Status feature! Instantly see which systems are connected, when they last synced, and whether any need attention—all from one place in DC305. See **Connection Status** to learn more.

New Feature: Find Help in DC305

Help is now built right into DC305! Click the new Help icons throughout the program to open the exact help page for the screen you're on. Get answers instantly without leaving your workflow. See **Find Help in DC305** to learn more.

We added a new help topic, **Find Help in DC305**, which shows how you can get answers faster with the new Help icons built into DC305. Instead of searching through the full help system, you can now open the exact help page for the screen you're on, navigate recently viewed topics, and still access the full Help library when you need it.

New Feature: Pen Inventory

The **Pens** topic now covers Pen Inventory, a new feature that tracks daily pen counts and trends automatically. Use it to monitor pen occupancy over time, adjust capacity, and easily export historical data for analysis. See **View Pen Inventory** to learn more.

Base Program

- A new **Pen Inventory** feature has been added, allowing you to track occupancy trends using a historical record of pen counts. The Pen Inventory feature is accessed via the `Alter\4` command. Please note the following:
  - Pen counts began logging in Version 25.7.1156.
    - The pen count is logged once per day, just after 12 am. Pen counts are logged as long as DC305 is running.
    - PEN=0 pen counts started logging in this release (25.9).
  - Pen count data is stored for two years.
  - The Pen Inventory feature's *Select Pens* function maintains your last selection when you export the .csv file.
-
- Removed Beef designation for existing Red Dane (RD) sires during Sire Cleanup.
- Fixed Beef tag series issues.
- The **Events\2 report (List cows and events)** has been enhanced to include ETDAM and ET Dam Registration for breeding events that include the information in the event details.
- Items for Last Parlor Visit Data and Last Parlor Visit Shift are now auto-mapped to the standard items.
- Fixed an issue when running a command and the "Total Number Entered" message in DC305 was not displayed.
- Updated 595H to report as conventional semen.
- Fixed an issue where ETBRED events were subtracting semen usage from semen inventory.
- Fixed an issue where EGRAPH was not respecting the `BY PN` (Previous Pen) statement. (Numerous factors can change the PN value, so note that the PN value is the value *as of today*.)
- In Version 10 of the cowfile, we fixed an issue where the protocol number changed unexpectedly when editing an animal's mastitis event remark on the DC Client, specifically when there were more than 255 treatment protocols. This occurred when the event was processed on the DC Server.

- Russian translations in DC305 have been updated.
- For Canada, DC305 has been updated with the 2025 NAAB proof files.
- Additional standard items for genomic trait sync with PULSE integrations have been added:
    - GBDC: Body Size Composite
    - GCHON: Chondrodysplasia
    - GFTI: Holstein Fertility Index
    - GJH2: Status for Jersey Haplotype 2
    - GSDYS: Spinal Dysmyelination

Integrations

- Lely: Support has been added for XID between DC305 and Lely.
- Lely: DC305 no longer sends barn names when the animal name field is de-selected in integration settings.
- Lactanet: Added support for 6-digit ID and CHAIN numbers in the loading in DairyComp for Canada.
- Jantec: Fixed issues with loading milk shifts with Jantec.
- HeiferPro: When importing pregnancy results via milk and blood tests when animals already have a current PREG status in DC305, if additional reconfirms of pregnancy are in subsequent imports, the import will create a PREG event (not an OK event).

CowCare

- The database now refreshes after changing group premises.

Pocket CowCard

- Fixed an issue where DC305 was sending ??????? for Dairy name/Herdcode.

New FAQ: Resolve HELD Commands

We added a new help topic, **Resolve HELD Commands**, which explains what HELD commands are and how to manage them in DC305. The topic shows how to identify pending HELD commands and use the HELD Command Processor to edit, repost, remap events, relax birthdates, delete, or delay them. This helps users resolve transaction errors and keep herd records accurate.

New Help Topics: Events

We added three new help topics that explain how events are used in DC305:

- **Events** introduces the concept of events, explains how they differ from items, and shows how to view, enter, and manage them on the CowCard.
- **Event Definitions** serves as a reference guide to all standard and optional events, with descriptions of their use in DC305.
- **Events Command** explains how to analyze historical event data across your herd using both the Events Options menu and command line, including examples of advanced reports and switches.

DairyComp 305 - Version 25.8

Release Date: August 26, 2025

DC305 version 25.8 contains various bug fixes, customer-requested enhancements, and integration improvements for our integration partners, as well as many other improvements.

Expand the topics below to learn more about the updates in this release:

Base Program

- Loading tags into EID Manager is now faster.
- The ABS Global 929 stud code has been updated to the Sexed Male Beef semen type.
- DCClient will now reopen after updates are installed.
- DC305 has been updated with the August 2025 NAAB proof files for the US, UK, and Germany.

Integrations

- ScotEID: Sending births to ScotEID now includes sire registration, which means more complete and detailed records.
- Neogen: We have updated the Neogen DNN file import to correctly identify and import data using column names.
- SCR: We've fixed an issue that caused the duplicate item check report to print automatically when sending data to a partner.
- AFI: You can now identify and send the custom item for a secondary transponder, ensuring your data is more complete.
- Grimmius:
  - Grimmius can now post SOLD, DIED, Received, and Returned events to DC305. These new settings are located in a new Traceability section within the `IMPORTF\R` command.

  - To use the new SOLD event feature, you will need to select a Disposal Code that will be used for all incoming SOLD events from Grimmius.
  - When using the DIED event integration, please note that Grimmius will not provide a reason for the death.

CowCare

- Improvements have been made to the CowCare drug inventory, ensuring it works as expected for more reliable tracking.
- You now have the ability to remove a Premise in CowCare if it is no longer being used, helping you keep your records clean and organized.

Updated Help Topic: Edit CowCare Settings

In the **Edit CowCare Settings** topic, we updated the instructions for the **Inventory Settings** section. The revised text now explains how to use the new **Add** and **Remove** buttons to create, update, or delete group premises, replacing the older **Combine** button workflow. This makes it clearer how to group or ungroup premises when tracking inventory across multiple locations.

New Help Topic: ALTER\2: Item Definitions

We added a new help topic, **ALTER\2: Item Definitions**, which explains how to view, create, change, delete, and rename items in DC305. The topic also covers advanced options such as mapping the item table, displaying item usage, and printing or exporting item definitions.

DairyComp 305 - Version 25.7

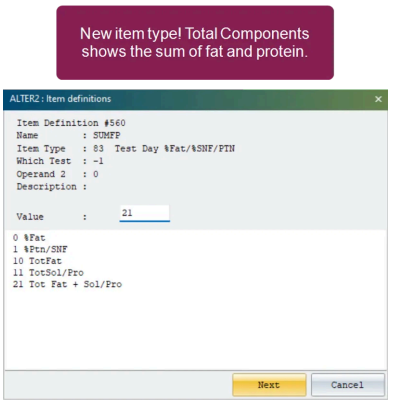
Release Date: July 29, 2025

DC305 version 25.7 contains various bug fixes, customer-requested enhancements, and integration improvements for our integration partners, as well as many other improvements.

Expand the topics below to learn more about the updates in this release:

Base Program

- Calf IDs now follow the correct ID pattern for Germany herds.
- Sire lookup can now properly handle Select Sires stud 585 sires.
- The Red Dane (RD) breed is no longer designated as a beef breed.
- The German HerdE conversion has been updated so all LACT=0 animals do not get a DNB event.
- Logon for DCTeach now opens with “Use Last Logon Date” by default.
- In SERVER\3, the reported IP address is now correct.
- DCCLIENT now updates when the only change on the server was an edit to existing commands.
- Item Type #83 has a new Op2 that can be used to show Total Components, the sum of fat and protein. Enter **21** for **Op2** to see Total Components.



Integrations

- Fusion: Added support for sending data to Fullwood Fusion-Crystal parlors.
- Genomics: When you remove a mapped selection, it now saves correctly in IMPORTF.

FTECH

All report options for Lactanet FTECH users are now available and fixed.

Updated Help Topic: View Lely Milk Production

In the **View Lely Milk Day Production** help topic, we updated the instructions to **Create Additional Lely Milk Data Items**. When you configure the new items, enter the values listed for **Value (Milking Day)** in the table to ensure that the milk data items work correctly.

DairyComp 305 - Version 25.6

Release Date: June 30, 2025

DC305 version 25.6 contains various bug fixes, customer-requested enhancements, and integration improvements for our integration partners, as well as many other improvements.

Expand the topics below to learn more about the updates in this release:

Base Program

- Implemented ET Bred improvements for embryo transfers entered in DC305 (excluding entries in the Embryo Transfer Module in PULSE). To learn more, contact **VAS Support**.
- Added `ECON\IH` (Heifer Inventory) to the command switch gallery.
- Fixed the item format in GRAPH when using Item Type #42.

Integrations

- Lely:
  - You can now **View Lely Milk Day Production** in DC305. See the help page to learn more.
  - DC305 no longer displays the DCSquirrel error that indicated that MSXML was not installed.
- Boumatic Milking Parlor: An exception error no longer locks up DC305 during MSEND.
- DairyMaster (MooMonitor): ABORT and OPEN events are now included in the data send for DairyMaster (MooMonitor).
- Germany: Reduced certain scenarios that produced incorrect alerts with `SEND\V` and `SEND\D` on German herds.
- DataFlow II (SCR): Adjusted alert messages that appear in DC305 during the data exchange with DataFlow II (SCR).

FTECH

Added FTECHx.def files to DCStartup and Web Updates.

New Help Topic: View Lely Milk Production

To allow producers to monitor this specific Lely-calculated performance metric, DC305 now imports and stores Lely milk day production values. Once your DC305 is updated to the relevant build *and configured for the Lely milk interface*, this feature will automatically appear on the **Lacations** tab of CowCards. See **View Lely Milk Day Production** to learn more.

New Help Topic: Analyze Dam Relative Value

The Dam's Relative Value (DRELV) item in DC305 allows you to evaluate your youngstock more precisely by incorporating the performance of their dams. While genomic testing offers genetic insights and parent averages consider pedigree, DRELV provides a direct measure of the dam's actual performance within your specific herd environment. See **Analyze Dam Relative Value** to learn more.

New Help Topic: Address Bad PLA Error

Have you ever received a bad PLA file error message? This error message is related to PULSE and prevents PULSE from properly updating with your DC305 data. See **Address Bad PLA Error** to learn more.

DairyComp 305 - Version 25.5

Release Date: May 28, 2025

DC305 version 25.5 contains various bug fixes, customer-requested enhancements, and integration improvements for our integration partners, as well as many other improvements.

Expand the topics below to learn more about the updates in this release:

Base Program

- Fixed a problem where the `ECON\SDCT` (Selective Dry Cow Therapy) command incorrectly set `SDCTC` for cowfiles that had French-translated event names.
- Resolved an issue where the `SETMILK` command wouldn't save the selection if a system item was used for Transponder Type.
- DC305 uses the German ECM formula in German cowfiles.
- Fixed a problem on DCClients where entering a FRESH event with a `PREV` event would cause `HELD` commands at the server.
- Resolved an issue where Item Type #10 was not populating correctly in `SUM` commands.
- Updated the send command for DairyTrace and SimpliTrace.
- Report names are now properly displayed if the command contains an `ENTER` command and then a `LIST` command.
- Fixed a problem where the REM picklist for the `FRESH` event was not populating for subsequent entries.
- Made an improvement so if an Item Type #190 uses the Sexed option, it now returns a default value of "-" if the item it references is blank.
- Resolved an issue with pen display that occurred when running parlor pen graph reports.
- Fixed a problem where the `SERVER\B` exception error caused DC305 to lock up.
- Updated DCSIRES with new sire proof data for Canada and Italy.

Integrations

- Lely: Fixed access table damage that occurred when entering FRESH events under certain conditions in DC305.
- Boumatic Milking Parlor: Fixed a problem where animal milk yields with an ID number of 9 characters were not loading into DC305.
- Genomics: Updated the GINB (gInbreeding) item to support negative numbers.
- DRPCs: Added two more health events to the SSF file: diarrhea events (DIAR) and digestion events (DIGE).
- HeiferPro: Fixed a problem where the `MEASURE` event would not save if it was mapped in `IMPORTF\HP`.
- Grimmius Import: Resolved an issue where the Grimmius import was creating duplicate events on the CowCard.

CowCare

- DCClient now properly calculates and displays protocols based on CowCare protocols.
- Added the ability to change the cow ID in unapproved (open) CowCare transactions. See **Updated Help Topic: Reassign or Delete Open Treatments in CowCare!** to learn more.

Updated Help Topic: Reassign or Delete Open Treatments in CowCare!

Sometimes, a cow's ID may change, or a treatment might have been mistakenly entered for an incorrect cow ID. CowCare provides a dedicated tool to manage this situation, allowing you to reassign treatments to a different cow ID or disable open treatments entirely. See **Reassign or Delete Open Treatments for a Cow ID** to learn more!

DairyComp 305 - Version 25.4

Release Date: April 23, 2025

DC305 version 25.4 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners, PCC, and many other improvements.

Expand the topics below to learn more about the updates in this release:

Base Program

- The `SERVER\H` command and the Server button can now be used on DCLIENTs to access the held command dialog.
- Starting with this release, new command abbreviations must be two or more characters long. Previously, command abbreviations with only one character caused issues if commands contained other single character items, like breeds. Existing command abbreviations with one character can still be used. This change only affects new commands.
- We've been increasing the number of sires in the sire files, so we made an improvement to speed up the sire selection list when using the SIRES command.
- Corrected an issue where the `SETUP\S` command would change the `VC=3` value to 40 if the **Days at Preg Exam** field was left blank.
- Fixed a problem that occurred when the wrong "Next Calf ID" was prompted on DCCLIENTs. This problem only existed in the German versions of FRESH.
- Resolved an issue with the Ukrainian translation file that caused an error in the milk imports.
- Fixed a problem where the EVENTS option to remove duplicate events would also offer to delete the event from the table.
- Resolved an issue that occurred during `GETCOW/PUTCOW\I` transfers where, in some cases, a previous lactation could overwrite the current lactation.
- Fixed a problem with incorrect sire name prompting in Russian language files.
- Updated the sexed sire table to support United Sires codes 596 and 796.
- Updated the sire files for the US, UK, and Germany with April 2025 proof data.

Integrations

- Lely: DC305 has implemented two new updates to ensure that the integration between DC305 and Lely Horizon is more seamless than ever:
  - DC305 can now detect and resolve animal identification issues used to map animals between DC305 and Lely Horizon.
  - If DC305 attempts to add or update an animal record in Lely Horizon and the update is not received, DC305 now tries to resolve the issue and resend the event data to complete the animal record in Lely.
- Beco: Fixed a problem where "No ID" cows would not show up on the parlor report.
- Germany: Resolved an issue with the GERMAN SEND command where it would not properly designate calvings that resulted from an embryo transplant.
- AFI Milk: When the AFI folder was offline, DC305 would lock up during the MSEND export. This situation is now resolved.
- Genomic Imports: We updated breed index calculations for Holstein Total Performance Index (TPI), Jersey Performance Index (JPI), and Jersey Udder Index (JUI).
- Milk Imports: We fixed a problem where the Last Parlor Visit date and times could be incorrectly defined.

CowCare

Fixed a CowCare-related error message at program startup.

FTECH

- Added a new report designation for Agsource FTECH.
- Implemented some rate changes for Minnesota FTECH.

PCC

We finally solved a longstanding problem where, in some cases, animals added on a PCC device connected to a DCCLIENT were not properly being added to the cowfile on the server.

DairyComp 305 - Version 25.3

Release Date: March 25, 2025

DC305 version 25.3 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners, as well as many other improvements.

Expand the topics below to learn more about the updates in this release:

Base Program

- Updated the sire files for UK and Germany.
- Added a safeguard to prevent an issue where animal records could be corrupted if you edited an event and DC305 processed incoming client transactions at the same time.
- Fixed a problem where redirected report outputs were not created properly if the option to disable all printing was enabled.
- We increased the number of sires in the USA sire file to support all sires born in the last 20 years, which is now 20,000 proven sires.
- Fixed an issue where the `SUM` command was not properly grouping if an Item Type #133 (Gender) item was used.

Integrations

- GEA DairyNet: Added ability to send 4-digit pen numbers to GEA DairyNet.
- NEDAP: The PULSE Nedap item for heat time (ACTIM) is now being properly created as an Item Type #23 (Date) item.
- All Parlors: We now create and store two new items:
  - Last Parlor Visit Date (LPVD): The date when the cow was last milked
  - Last Parlor Visit Shift (LPVS): The milking shift number for that date
    With these two items, along with the existing START (start time) item, you can now create reports to find cows missed for milking, the percentage of the milking herd that went through the parlor, etc.
- Germany: Fixed a problem where sometimes the test dates were not properly imported for a new herd created from German load files.

DairyComp 305 - Version 25.2

Release Date: February 26, 2025

DC305 version 25.2 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners.

Expand the topics below to learn more about the updates in this release:

Base Program

- Improved how quickly DCClient updates the expiration date after you update DC305.
- Corrected an issue where the `BREDSUM\S` command would incorrectly group sires if the main cowfile's ShortName setting did not match ARC file.
- DC305 now displays a warning message if you try to delete a FRESH event from an animal's CowCard when there are other events.
- Fixed the Average Daily Gains (ADG and ADGI) report display in Russian.
- Fixed an issue where DC305 incorrectly added Stud Codes 522 and 551 for foreign sires during sire cleanup.

Integrations

- Allflex: Fixed the Pregnant Open Date (PODAT) and PREG fields in the status file.
- Beco: Fixed an issue where DC305 showed an incorrect M1 milk value.
- Boumatic: Resolved an issue where an exception error would lock up DC305.

FTECH

The SSF import now manages duplicates better.

DairyComp 305 - Version 25.1

Release Date: January 29, 2025

DC305 version 25.1 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners.

Expand the topics below to learn more about the updates in this release:

Base Program

- Released the updated Canadian sire file.
- Released the updated Portuguese language file.
- Fixed an uncommon problem where the auto backup process would lock up DC305 if there were antivirus or external backup programs interfering with the backup file creation.
- Resolved an issue where the `BREDSUM\*` (By Sire Sexed Type) report incorrectly split the embryo results into separate lines.
- Added protective checks to prevent a client refresh from corrupting an animal record during a CowCard event edit.

Integrations

- Delpro: Delaval fixed a problem where milk weights were not loading properly.
- Grimmius: Resolved an issue where the Grimmius weight and measure import would sometimes create duplicate `MEASURE` events.
- Lactanet DairyTrace: Fixed an issue where export records in TRACE module were not properly implementing foreign and domestic tags.
- Lely:
  - Fixed a problem detecting future transfer dates that affected Horizon.
  - Resolved an issue with Lely animal ID assignments that occurred when an import was done on a client.
  - Fixed a problem where, in some cases, the animal group numbers on Lely were incorrectly changed by the `MSEND` export.
- SCR: Previously, the `MSEND` export to SCR was in the report browser when duplicate tags (HTTAG) were detected. This has been changed to create an alert instead of a report.
- Export (UK): Made a modification so that the UK `RECEIVE` command can now recognize some occasional format errors in the files provided to DC305.

FTECH

We now allow IDs of up to 8 digits for the FTECH Cow Transfers records.

Pocket CowCard

- Modified the logic for handling the default assignable items that can be used in PCC scanning, primarily avoiding the default use of item 4 (usually `REG`) for scanning lists sent to PCC.
- Added a new option to quickly process pregnancy check vet codes in PCC. When you define some vet codes in DC305, you can now check the PCC**:** **Auto Preg Event** box. After you enable the option, you can easily process an animal on some worklists by simply tapping and scanning an animal. This feature requires the latest version of PCC.

DairyComp 305 - Version 24.12

Release Date: January 8, 2025

DC305 version 24.12 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have implemented numerous improvements related to integrations and traceability systems.

Expand the topics below to learn more about the updates in this release:

Base Program

- For customers who order match pair tags directly from VAS, DC305 can now automatically reorder new EID tags when it detects that you only have a few EID tags left. To enable this feature, enter `ALTER` in the command line and open the EID Manager, then click **Enable** at the bottom right side of the screen. After you set the number of EID tags and click **Join Program**, a VAS sales representative will contact you whenever your inventory falls below the set number. You can edit the settings any time in the EID Manager. See **DairyComp EID Tag Manager** on the VAS website to learn more!
- Fixed an issue where DC305 would incorrectly assign new events to technician teams that were just created.
- Corrected a problem where `ECON\H` would sometimes report the wrong gender for calves produced by embryo transfer in the Holstein ERA file export.
- You can no longer use `EVENTS\8M` to erase events in batch. This feature is reserved for the VAS Support team.
- Removed the ability to enter a blank date in `VENTER`. Entering a blank date would cause errors in subsequent entries.
- Corrected the default withdrawal lengths for the DryClox and Spectramast protocols.
- Added an option in **SETUP** > **Printer** to disable all printing. This can be used for server setups where most of the work is done on clients and the server machine is unattended.
- Released a new USA sire file with December 2024 proofs.
- Fixed an issue where the `EVENTS\2` event listing report did not display technicians higher than 255.
- Corrected a problem where archive files weren't always refreshing to clients.

Integrations

- Brazil: Updated the Portuguese language file to fix a display issue for test days.
- DairyMaster: Added support for the new version of DairyMaster: DairyVue360. To enable this, add the `\V2` switch for the new version.
- HeiferPro: We now skip incoming SOLD/DIED entries from HeiferPro if they were already entered manually in DC305.
- Lactanet: Fixed a locked file problem that occurred when a Lact-T export file was written onto a USB drive.
- Lactanet DairyTrace: Fixed the foreign indicator for import and export events in DairyTrace.
- Minnesota DHIA: Made some requested modifications for the FTECH lab report headers.
- NEDAP (Velos): Extended the allowable ID length for the NEDAP Velos integration from 5 to 8 digits.
- SEND (Germany): Made requested changes for the German SEND export command (ADS files on test day).
- UK ScotEID Traceability: The ScotMoves SEND command can now be scheduled in the task scheduler.

DairyComp 305 - Version 24.11

Release Date: November 19, 2024

DC305 version 24.11 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future releases.

Expand the topics below to learn more about the updates in this release:

Base Program

- For herds that use sire short name entry, we now display the sexed code (if present) in the sire selection picklist in **BRED**. This makes it easier to identify and choose between the sexed and unsexed named sires. For example, we show the sexed "TROY" as "TROY.S" to denote its name and its sexed code.
- For animals with a reproduction code (RC) of zero, we used to display a blank space for the RC item. We now show "HEIFER" (translated to the local language) for these animals. The **SETUP** command can be used to change how RC is shown for your animals.
- Updated the **BREDSUM Options** dialog to support several of the newer switches available in DC305.
- Added a new Item Type #174 to show embryo transfer pregnancies. If the **Op1** is `0`, DC305 returns a "Y" or "N" to show whether the animal is currently pregnant from an embryo transfer implant. If **Op2** is a `1`, it shows the PULSE Embryo ID of that implant (if the embryo transfer was entered in PULSE).
- For the `BREDSUM\*` (BREDSUM by sexed code) option, we now display embryo implants as a separate category.
- Fixed an issue where the EGRAPH event `\P` was not properly displaying protocol numbers over 255.
- We now have a secondary sire ID available for some regions that do not support NAAB codes. When you define an Item Type #190 item, you can now select a NAAB code as a **Secondary Sire ID**. For example, in Germany, the DHV codes are the primary sire identification and you can use this new item to show the NAAB code for that sire. Note that you need a V10 cowfile to enable this feature.
- In some cases, if DC305 found a change to the sire ID during a genomics file import, it would change the sire ID to an ISO number, such as 84000012345567. We added safeguards to prevent unwanted changes like that. Additionally, we now check against the global sire reference file (V10 only) for improved sire ID checks and corrections when possible.

Integrations

- AFI Milk: The AFI milk import now supports pen numbers over 255. Note that not all parlors allow this. Use caution when creating pen numbers not supported by the partner software!
- Germany: Handled requested modifications in `SEND\V` testday export: the MLP.ADS file for all LKV-Codes.
- Germany: You can now run KuhVision export (`SEND\K`) from an automated scheduled task.
- Lely: For the past few months, we've been field testing the change to use the Lely Animal ID as the primary ID for the Lely integration. It has been successful, solves multiple problems, and, with this version, it is now the default.
- Zoetis Genomics: Updated the formula used for TPI calculations in the Zoetis DNG/DND genomics file import.
- Zoetis Genomics: We now allow defining an alternate item to be used as the lookup when importing Zoetis Genomic DND files. Note that this is not needed in most cases (and not in the US), but it was required for some foreign countries.

FTECH

- MN FTECH requested a billing change (% field tech vs. association).

DairyComp 305 - Version 24.10

Release Date: October 29, 2024

DC305 version 24.10 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future releases.

Expand the topics below to learn more about the updates in this release:

Base Program

- To speed up client refresh speeds for large herds and those with poor connections, archive files are only refreshed to clients if they have been updated from a `CLEANUP`.
- Added support for the `PCT` command to display values for Item Type #145 (Tech name) items, e.g., `PCT LACT=1 BY MTECH`.
- The `ALTER` Protocols dialog was adjusted to properly limit maximum values for the protocol withdrawal date.
- Fixed an issue where event edits for BRED Tech completed on DCCLIENT were not always sent correctly to the server.
- Resolved an issue where editing Combo events was not working properly in some situations.
- The Client Timeout value was previously limited to 20 seconds on the DC-Shadow version. It was increased to support larger values for large herds and/or poor connections.
- Fixed an issue where the Average Daily Gain (ADG) at Age calculation could return incorrect values for weights in kilos.
- Fixed an error message shown in BRED when the SIR5 (use 5 extra sires) option was enabled.

Integrations

- Lely: Fixed a problem where animal IDs of zero could be sent to Lely.
- Neogen: Added a new Holstein haplotype, Polled Mutation Trait (GHP), for Neogen genomic import.
- Lactanet: Released updated sire file for Lactanet.
- Zoetis: Added the ability to use an alternate lookup ID when importing Zoetis Genomic files. This is only needed for non-US customers.
- Universal: Added the ability to define a different folder for the `MSEND` command for Universal TAUPRO transfer program.

CowCare

- Added the ability to duplicate drugs for a new premise.
- Fixed a problem that occurred when editing usages.

PULSE

Resolved an issue where protocol changes completed on PULSE were not immediately applied in DC305.

DairyComp 305 - Version 24.9

Release Date: September 24, 2024

DC305 version 24.9 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future releases.

Expand the topics below to learn more about the updates in this release:

Base Program

- We've dramatically improved the speed of the `CLEANUP` command whether you use it with the setting to remove duplicate records or you use `CLEANUP\A` directly.
- Added support for `SIR5` and `SIR6` items, which can be used for BRED event entry with default recommended service sires, along with the existing `SIR1` through `SIR4` items.
- Released the August 2024 USA sire proofs file.
- In DC-Shadow refresh options, you can now increase how long a refresh runs before it times out and fails. Extra time is sometimes needed for systems with slow internet connections.
- Fixed a problem where `LOGON` would lock up when a corrupted cowfile existed in the multiherd folder.
- Resolved a performance issue where viewing multiple CowCards was getting slow. This was caused by an accumulation of a large amount of sensor activity data.
- In order to allow genomic imports in some foreign countries (including Argentina) with different government ID systems, we now allow changing the default item used to look up the cows from the DNG file.
- Fixed an issue where the system would not correctly delete a third calf record if you deleted a FRESH event with three live calves.
- Resolved an issue where `FILEOUT` did not always export the expected set of items. This was a rare condition related to the milking interface in use.
- Fixed an issue where the `EVENTS\3` Calf Report was not showing all of the expected items.

Integrations

- NEDAP Now: Added support for using activity-level data received from PULSE in the `BREDSUM\J` activity summary.
- Cow Manager: Added support to send PREV events to Cow Manager in the same way we send PREG events.
- Jantec Milk Import: We now detect and accommodate for a date bug in the Jantec milk file, which caused milking #3 to be missed if it was scheduled after midnight. The weights now load as expected.
- Protein Serum Import: Added support to automatically import protein serum lab result files (DNY files).
- Lactanet Traceability: Fixed a problem designating domestic versus foreign exported cows.
- SEND (Germany): We now allow users to set different premises to different emails to differentiate between breeds.
- GEA DairyNet: Added support for GEA's new DairyNet integration.

CowCare

Fixed an issue where CowCare would freeze when a very large number of backlogged drug usages were processed.

Expand the topics below to learn more about new online help content!

New Help Topic: Sign Up for DC305 Training Classes

Interested in brushing up on your general DC305 knowledge or learning more about specific DC305 strengths, like the power of its reproduction analysis tools? VAS offers several instructor-led and self-paced courses! See **Sign Up for DC305 Training Classes** to learn more.

DairyComp 305 - Version 24.8

Release Date: August 20, 2024

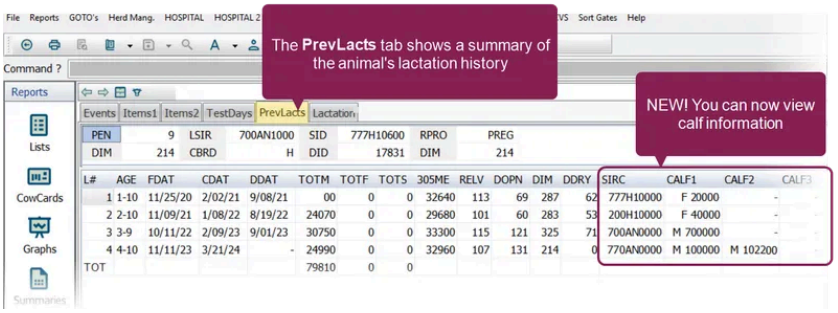
DC305 version 24.8 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future

releases. One highlight in this release is that you can now view calf information in the **Previous Lactations Tab** of any animal's CowCard.

Expand the topics below to learn more about the updates in this release:

Base Program

- The CowCard's **Previous Lactations Tab** now includes calf information: the calf's ID, sex, status, and sire. These will be automatically backfilled the first time you run DairyComp version 24.8:



- The USA sire file now contains sires active in the last ten years.
- Added translation capability to the `ECON\IR` report header.
- Fixed the problem where a client refresh would occasionally erase the local client task table.
- For very large herds with a lot of activity logging, we now properly handle any overflows when the log file gets too large.
- The OTF (on-the-fly) GUIDE items are now always initialized by default for DairyComp Consultant users. Additionally, you can force loading of the OTF items by using `ALTER\20`.
- Bull stud code 694 is now properly recognized as sexed female semen.
- When a sire is renamed in ALTER8 : Sire Table, the sire is also renamed in any archive files. This improves BREDSUM by sire.
- On DCClient, the previous release inadvertently removed access to some `ALTER\11` capabilities. This has been restored.
- OTF items can now be used in EVENTS HDR files.

Integrations

- Lactanet: Added the **REGID** field to the XML file created using `ECON\LACT`.
- SEND (Germany): Added support for sending ADS files for multiple AEs (premises) to different email addresses.
- Zoetis Genomics: Fixed an issue where DNG files from Zoetis were not imported correctly because the files were in an unexpected format.

PULSE

- DC305 now accepts breeding codes entered on PULSE even if the breeding codes are not enabled in DC305.
- Resolved an issue where events were sometimes duplicated in DC305 if the event was entered in PULSE with lowercase remarks.

DairyComp 305 - Version 24.7

Release Date: July 23, 2024

DC305 version 24.7 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future releases. This release had a larger focus on accommodating customer requests.

Expand the topics below to learn more about the updates in this release:

Base Program

- The `SIRES\M` command, which is used to import sire matings from a file, now supports SIR4 and SIR5.
- Improved Item Type #88 to return the number of test days using `log SCC > [value]`. Op1 should be set to `97`, and Op2 should be set to the cutoff value for log SCC. For example, If you want the count of test days where the log SCC is over four, enter `4` for the Op2 value.
- We now only display pens in remarks up to 3 characters unless 4 characters are required. In Cowfile Version 10, we allow up to 9999 pens and we changed the remark of MOVE events to support 4 characters for each pen. This caused remark lengths to exceed 8 characters, which reduced readability in some cases.
- Added content to the `ALTER` routines, which are now visible when using DCClient. While they are still read-only (except for scheduled tasks), you can now view the pens, sires, vet codes, and events tables.
- Added updated sire files for the US, Canada, and Italy.
- Fixed a small issue where reproduction codes were not recalculated properly when a BRED was entered on the same day as an ABORT event.
- The Switch Manager, which displays all available command switches, can now be accessed from the Help menu.
- Improved the way DCCLIENT is updated, which fixes some instances where the DCSquirrel utility tool did not update properly.
- Fixed a typo in the switch description for logging onto archive and backup files.
- Updated the standard event code names for French, Italian, German and Russian users.
- Added Speckled Park (SK) as a beef breed.
- Fixed a problem where CAR codes defined on the server were not being sent properly to the client. Due to the variation in how DCSETUP files are handled between the server and client, we added a setting in the SERVER configuration message that forces the transfer: “Send custom CarCode+VEnter Keys to clients.”

- Resolved a problem where the system would modify FRESH table items when one of the items in the paired tables was deleted.
- Improved the client refresh process, which resolves an issue where clients could have lost their local custom task scheduler tables.
- Reverted a change that caused an issue where a reproduction code was not sent correctly during the iTrans process. The change caused a problem with identifying male calves.
- Fixed an issue where changes made in the last update could have caused a failure in the Lameness Manager, depending on your PULSE settings.

Integrations

- Lely: The latest conductivity values can be imported from Lely in the daily milk import. To enable, use `SETMILK\INSTALL` for Lely and click the button to enable and create four teat items: CNDLF, CNDRF, CNDLR, and CNDRR. The items will start being updated in the milk weight loads.
- Lactanet: Added the **REGID** field to the Lactanet ECON export for Eastern Canada.
- CattleEye: Fixed a problem where values sent from PULSE were not displayed in the correct user-selected format.
- SEND (Germany): Fixed an issue with the German SEND command where MOVE events that involved outside premises were not sent properly.
- DNG Genomic Import: Added HPP (Holstein Horn/Polled) to items that can be imported from genomic files.
- Holstein EASY: Added support for all of the available items in the Holstein EASY import (`ECON\H`). Previously, we just supported importing the CLASS item.

CowCare

The CowCare basis drug file has been updated with new drug information; this happens once a year.

DairyComp 305 - Version 24.6

Release Date: June 26, 2024

DC305 version 24.6 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future releases, as well as completed some international requirements. The next release will include more base program improvements.

Expand the topics below to learn more about the updates in this release:

Base Program

- DC305 lets you customize event names, but sometimes, that makes it difficult for the system to recognize what the customized event name is supposed to represent. We implemented some standardization for events to improve event analysis in both DC305 and PULSE, especially for the PULSE features, **HealthVal** and **OneView**. You don't need to change anything, but to improve the system's analyses, we now have an option for every event in the ALTER - Events dialog where you can select the standard event description.
- Fixed an issue where Item Type #190 Breed item for some 501XD NAAB codes was not always properly seen as "XD".
- Updated the **Switch Manager** to include switches for the SSF SEND command (`SEND\DNL`).
- Implemented full support for Ukrainian language translations, including user-defined events and items.
- For international users where sire NAAB is not available, you can now enter the sexed code and breed for that sire when you enter a BRED event and add a new sire.
- GUIDE now properly reloads if you log on to a dairy with a different GUIDE file.

Integrations

- SEND (Germany): Added support for maintaining a description for each premise in the German premise table.
- SCR Milk Load: Fixed an issue where the system would double up milk weights of a milk load twice for the same milking.
- Delpro: Fixed a problem where a milking time set to midnight might result in an error message.
- FTECH (DairyOne): Added support for IDs that are greater than 5 digits in the lab data merge file.
- FTECH (Minnesota): Fixed a minor display issue in billing report.

CowCare

The "Drug Usage by Cow ID" CowCare Report now allows entry of a list of cow IDs to show on the report.

PCC

We recently changed incoming PCC event entries to use the same command as defined in DairyComp. However, we fixed an issue where the system did not correctly process items in an event command if the items were listed before `EC`. For example, if you enter a **MOVE** command, `PN=PEN EC=9 EDAY REM`, the system now sets the `PN` item properly before the system processes the **MOVE** command (`EC=9`).

DairyComp 305 - Version 24.5

Release Date: May 21, 2024

DC305 version 24.5 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future releases.

Expand the topics below to learn more about the updates in this release:

Base Program

- Fixed a problem that occurred when heifer weights imported on a client were sent to the server.
- Added Ukranian language support.
- Updated the field names in ECON reports to clarify that the **Totals** shown are **Total Animals** in the count column and **Total CWT** in the milk production column.
- Restored the ability to add days to protocol details using ranges (like "2-4" for days 2 through 4).
- The system no longer allows entering DRY events for heifers because it creates a misleading reproduction status and causes problems later in the process. These instances should be entered as treatments with an optional protocol rather than as DRY events.
- Released the latest sire files for the UK and Germany.
- Updated iTrans heifer files (iLoop DNT transfer files) to automatically exclude the reproduction code item, which resolves problems on the receiving side.
- Fixed a problem that occurred when entering sexed sires if the sire's only identification was an ISO code, such as in Italy.
- Fixed Item Type #73 items for "Pen from MOVE event" that broke in a previous release.

Integrations

- Preg Results Lab Import: The system now displays an alert and does not import lab-generated PREG events that occur after an ABORT event if there are no recorded breeding events since the abort.
- FTECH: Fixed the FTECH ABSORB command, so you can now import milk weights for cows with IDs over 5 digits long.
- SEND (Germany): Added several requested improvements to the German SEND command.
- SEND (Italy): Implemented the ability to add a designated EID item to the Italian SEND command. You can now use `INVIA\X [itemname]`, so in this example, you can use `INVIA\X EID`.
- SEND (UK): By request, Northern Ireland bulls are now excluded from the UK checksum validation process.
- Lactanet Traceability: Modified the display of the entry dialogs as requested by Lactanet.

CowCare

Added a new DrugWatch Cow History Treatment report to CowCare, which replicates the output from the former DrugWatch application. To learn more about viewing CowCare reports, see **View Reports**.

PULSE

DC305 now displays notification bars on the main screen if you need to update DC305 so it can properly synchronize with PULSE. Follow the instructions in **Update DC305 to the Latest Release** if you see the notification.

DairyComp 305 - Version 24.4

Release Date: April 23, 2024

DC305 version 24.4 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future releases.

Expand the topics below to learn more about the updates in this release:

Base Program

- You can now use the Switch Manager to see all the available switches for all commands in DC305! Simply click **Command?** at the left side of the command line to open the Switch Manager, then browse the list of commands and switches.

  See **Command Line** to learn more about using switches in the command line. For a comprehensive list of commands and switches you can download, see **Command Line Quick Reference Sheets**.
- The April 2024 USA sire file has been released.
- Fixed a problem that occurred when displaying cow pictures in the CowCard's Picture tab.
- Added language support for Estonia.
- When you export a report as a CSV file, the exported version is now easier to read because it no longer includes the Totals line.
- When you use the ALTER command, you can now enter a new technician number using the keypad.

Integrations

- Lactanet Genomics: Fixed an issue where the Lactanet genomics file import changed some sexed sires to their unsexed variant. This occurred because the unsexed sire NAAB was listed in the DNV file.
- SEND (Germany): Implemented more requested changes to the German SEND export format.
- BREDSUM (Italy): Fixed an issue where `BREDSUM\M` was not properly showing sexed sires with Italian stud codes. Also, in some cases, the command would lock up for some Italian v10 cowfiles.
- SEND (Italy): The Italian SEND command now supports electronic IDs.
- Boumatic MSEND: Resolved the error message caused by corrupted files from Boumatic.
- Morocco: Morocco can now use short sire names.

CowCare

Added a new CowCare report called **DrugWatch Cow History Treatment**, which replicates the output from the former DrugWatch application.

PULSE

DC305 now displays notification bars on the main screen if you need to update DC305 so it can properly synchronize with PULSE. Follow the instructions in **Update DC305 to the Latest Release** if you see the notification.

Pocket CowCard

- Fixed a small problem that occured when merging some PCC expert batch commands into the DC305 event command.
- Fixed an issue where the previous Device ID was still being shown on subsequent transactions in the Activity Log.

DairyComp 305 - Version 24.3

Release Date: March 26, 2024

DC305 version 24.3 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future releases.

Expand the topics below to learn more about the updates in this release:

Base Program

- The academic build (DCTEACH) has been updated to read Cowfile Version 10.
- Fixed a problem where, in some cases, if you entered a `BRED` event as an embryo transfer, the CowCard did not properly display `ETBRED`.
- In Cowfile Version 10, the `ALTER` dialog can now support up to 512 breeding technicians. Previously, the `ALTER` dialog only supported 250 technicians.
- Fixed an issue where the CAR code table did not refresh properly for shadow clients.
- Fixed a problem where the system would incorrectly group the crosstab, `BREDSUM\XC BY LCTGP`, by breeding technician.
- If you enter a `BRED` event for foreign herds that use short sire names instead of NAAB codes, DC305 now displays the sexed code for the bull. Additionally, when you add a new sire, you can now enter the bull's sexed code.
- Fixed an issue where the protocol remark was being truncated when exporting culture results to QSM files.
- Improved the search function for sires so that you can enter or find a sire using either the default name (usually NAAB code) or the ISO tag. This works for LIST and SHOW, as well as sire imports.
- In Cowfile Version 10, you can now enter up to 2048 protocol details.
- Fixed a problem where the system would not accept stud codes over 255 in the `SIRES\8` analysis.
- Fixed an issue where the system did not properly update downloaded pedigree (DNZ) files.
- You can now limit access for modifying and recalculating the MONITOR in the User Manager.

Integrations

- UK SEND: Fixed an issue where the `SEND` for the UK BCMS system did not properly default to the latest send date.
- German SEND:
  - Implemented requested changes for sending the milking frequency.
  - Implemented requested changes in the MLP and KARIND export files.
  - When importing the Lactanet DNV genomic file, it will update the animal sire and dam registration if the DNV file has different information than the cowfile.
- Minnesota DHI: Implemented requested change for the MNDHIA FTECH billing module.
- Lactanet Genomics: Implemented requested changes for `MOVE` and `SOLD` events in the KARIND export file.

CowCare

- Updated the default withdrawal period for some drugs in the default protocol list, which is used when adding a new protocol. Existing protocols defined with those drugs are corrected.
- CowCare now defaults to the latest (Version 2) behavior.

PULSE

If you enter an invalid `FRESH` event in PULSE and send it to DC305, DC305 now creates a held command for the event so you can review and verify the event details. For example, DC305 might create a held command for a `FRESH` event if you try to enter it just a few days after the previous fresh date.

Pocket CowCard

DC305 now merges event entries from PCC with the existing event commands to support any items defined for that event. For example, if your `DRY` event is `EC=11 PEN=5`, a `DRY` event from PCC will now support the `PEN=5` entry.

DairyComp 305 - Version 24.2

Release Date: February 20, 2024

DC305 version 24.2 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future releases.

Expand the topics below to learn more about the updates in this release:

Base Program

- Corrected an issue where the CowCard incorrectly displayed protocols on BULLPEN events.
- Posted the Italy sire file with December 2023 proofs.
- Fixed V9 issues where protocols above 255 did not work properly in the previous update and high-numbered protocols were not transferred correctly from DCCLIENT.
- Resolved an issue where `FILEIN` might not import the last record in a file if the file contained blank values after the last line.
- Fixed a `BREDSUM` problem where grouping bull studs did not work correctly wherever NAAB codes are not used, such as Italy.
- Some users reported missing IDs from the `CHKFILE`: "List unused ID numbers." This occurs if the IDs are being used in the EID inventory. To exclude EID inventory IDs from DC305's ID check, check the **Ignore IDs in EID file** box.
- In Parlor reports, we now implement a tolerance value based on parlor size, which reduces the number of false suspect stalls in the report.

Integrations

- Minnesota DHI: The SSF `SEND` command for Minnesota FTECH will export health events by default. For other DHIs, you can add the `\H` switch (`SEND\DNLH`) to send health events.
- Allflex API: Fixed a problem where the system could create new items that overlapped existing items during SETMILK installation for the Allflex API.
- NEDAP & BECO: Added the ability to use both NEDAP and BECO transponders on a single DCCLIENT system.
- ScotEID: Improved the date prompt and the Last Send Date default in ScotEID.
- Lactanet RECEIVE: Fixed an issue where system did not convert some of the lactation totals and projections in the Canadian RECEIVE for herds that use pounds instead of kilos.
- Lactanet Genomics: We now support the import of the Canadian format genomics files, DNV files, which are created and distributed by Lactanet. You can use the command `IMPORTF\V` to import them.

CowCare

- Fixed an issue where the system did not properly display event technician names in the CowCare usage summary.
- Fixed a problem where the system would print the wrong table.

PULSE

Fixed a problem where the system could might not load animals with a missing previous lactation in the initial PULSE load.

DairyComp 305 - Version 24.1

Release Date: January 23, 2024

DC305 version 24.1 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. We have also implemented numerous internal improvements, which will become evident in future releases.

Expand the topics below to learn more about the updates in this release:

Base Program

- Fixed a problem where, in some cases, the system would lose data about the previous lactation if you used `RESTORE` on a deleted cow.
- You can now configure DCCLIENT to create automatic local backups. If you want to enable this feature, open the **Client Configuration** menu and check the box next to the option.
- Resolved an issue where the system would display an error message ("Cannot find Cowfile.ARC") if you ran the BREDSUM by Tech report, then clicked on a technician.
- Improved `SIRES\9` so that it can now be used to update sire items for dead cows, as well as live cows (e.g. `SIRES\9 PL:PL FOR ID>0`).
- Fixed an issue where some transactions from Chinese DCCLIENTs would not process correctly on the server.
- Added the Holstein Muscle Weakness trait (GHMW) and Brown Swiss BH14 haplotype (GBH14) to the genomics import.
- If herds use sire short names as the default, the system now displays semen type of the sire (e.g. Sexed, Beef, etc.) in the picklist for `BRED` or `SEMINV` commands.
- Added the Selective Dry Cow Therapy GUIDE tab to the Russian GUIDE.
- When EPLOT was used to load milk weights on a client, the system would show "Cow Not Found" held command errors on the server. These have been changed to **Activity Log** messages.
- Fixed the header column alignment for the `BREDSUM\R` report.
- Fixed an issue where if you selected **Help** > **About** from the toolbar while the internet was disconnected, the program could temporarily freeze.
- Posted the USA sire file with December 2023 proofs.
- Posted the Canadian sire file with December 2023 proofs.
- Fixed instances where a previously-used protocol number was incorrectly used in subsequent `ENTER` commands.
- Resolved an issue where the system did not properly apply hospital pens to protocols when the pen number was over 999.

- Fixed an issue where the system did not properly input enter spaces in the protocol detail dialog.
- Resolved an instance where combo events for V9 included an erroneous event 72.
- Fixed an issue where the Previous Pen item was not working properly for pens numbered higher than 255.

Integrations

- Czech Republic: Released a traceability/government reporting system for the Czech Republic. This is in early beta testing at this time.
- Heifer Pro: Fixed a problem with sires in the HeiferPro import, which was caused by the inclusion of sire names in the export file.
- MN FTECH: Modify the configuration for the MNDHI Degree of Difficulty entry.
- Germany SEND: Requested a change for `SEND\D` commands to properly designate `SOLD` and `CAR` code usages.
- Delpro: The "Suspect milk deviation" was not being properly set in the `SETMILK` configuration dialog for Delpro. This has been fixed.
- Delpro: Resolved an issue where the stall numbers displayed in the Delpro Rotary Parlor report were off by one.
- Dairy One FTECH: Fixed the `SEND` command for Dairy One, which did not support more than 32,000 cows in the cowfile.
- Jersey Genomics: Updated the formulas for the Jersey Performance Index (JPI) and Jersey Udder Index (JUI) calculations.
- SCR/Allflex: Removed the alert that was posted if a cow ID of zero was in the import file.

CowCare

Moved all remaining CowCare version 1.0 customers to CowCare version 2.0. See **CowCare** to learn more about using the new version of CowCare!

PULSE

Fixed a problem where `SOLD` / `DIED` events were not properly removed from PULSE if a cow was mistakenly sold and then restored in DC305.

Pocket CowCard

- Resolved an issue where the system did not properly send protocols to PCC for V9 herds in DairyComp version 23.12.
- We now send more details about Source Farm names and IDs to PCC.

DairyComp 305 - Version 23.11

Release Date: November 28, 2023

DC305 version 23.11 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners. The majority of our time has been spent on groundwork for new features that will be released over the next two versions.

Expand the topics below to learn more about the updates in this release:

Base Program

- Fixed an issue where the system wouldn't correctly send FRESH technician data to the server.
- Added a check so the system does not resend DNC files (from `PUTCOW\I`) that are older than 7 days.
- When OTF ("on-the-fly" GUIDE items) are imported, the system now displays a message in the **Activity Log**.
- Added a new message that warns you if you try to enter a date for a SOLD or DIED event that occurs before the date of the latest event.
- Removed the extra technician number that sometimes displayed in the Activity Log for BRED events.
- Fixed a problem where the expected milk value was displaying the average instead in CowCards and items.
- Due to the increasing number of ten-character NAAB codes, we expanded the width of sire item displays so the codes are not truncated.

Integrations

- Agsource FTECH: Fixed an issue where dairies with over 32,000 animals were receiving an error if they entered `SEND\L` for FTECH.
- ScotEID: Corrected an issue where the ScotEID traceability was unexpectedly changing the EDAT for some cows.
- Neogen: Added a small change to the mail used for Neogen genomic submission files.
- UDS: Added some items to support the UDS Alta/Genex integration: Female Index, Female Rank, and Progeny Index.
- MNDHIA: We now always send health events in the MN version of the SSF SEND.
- Lactanet DairyTrace: We no longer remove MOVEIN traceability records with errors until they are manually resolved.
- UK SEND: We handled and reported internet connections errors more gracefully.
- MN FTECH: Properly imported the DBRD and CBRD values from PC-DART file.
- German SEND: Modified `\M` switch behavior for sending uploaded ADS files to German LKVs.

CowCare

- Fixed an issue where newly created dairies could not override drug quantities.
- Added an option to merge drugs together to reduce duplicate usage records.

PULSE

We now send the embryo ID and embryo code from BRED events to PULSE so PULSE can create the related type 77 items.

DairyComp 305 - Version 23.10

Release Date: October 24, 2023

DC305 version 23.10 contains various bug fixes, customer-requested enhancements, and integration improvements for PULSE and our integration partners.

Expand the topics below to learn more about the updates in this release:

Base Program

- Fixed a problem with the French translation of the OK event.
- Removed the restriction that blocked client refreshes if held commands were still present on the server.
- Added support for some custom mastitis/SCC analysis items that are used internally by GUIDE.
- When a protocol is edited in `ALTER\7`, it is now properly logged as an ALTER note in the **Activity Log**.
- We increased the number of items that can be added to the user-defined main menu.
- The technician table was adjusted to ensure incorrect technicians are no longer included in ALTER.
- Currency items with Item Type #101 (item + item) now display properly.
- Select Sires stud code 585 is now assigned to stud code 14 for sire look-up purposes.
- Fixed a problem where, in some cases, the system would overwrite the local client task table with the server task table if a client refresh failed.
- The type 190 sexed/breed display now shows an **R** instead of a **P** for beef sexed female sires. See **Using Sexed Sires** to learn more.
- Improved the display of `BREDSUM\EY` for large herds by increasing the column width.

Integrations

- Nedap: Added support for configuring Nedap Now alert items for PULSE. See **Step 2 in Nedap's Activate Integration topic** in the VAS PULSE Platform online help for item details.
- Lely Horizon: Unexpected PREG events are no longer sent to Lely Horizon.
- SETMILK: The SETMILK installer no longer displays shortened field names.
- BMOI: Fixed an issue where MOVEIN events were missing the declaration date for the Simplitrace traceability system.
- Lactanet Traceability: Additional items are now included in DairyTrace MOVEIN and MOVEOUT events.

DairyComp 305 - Version 23.9

Release Date: September 26, 2023

DC305 version 23.9 contains the usual assortment of bug fixes, customer-requested enhancements, and integration improvements for DairyComp, PULSE, and our integration partners. This update includes base program improvements such as sexed sire handling, data integrity and validation improvements, as well as continued work on integrations with our vendors and PULSE.

Expand the topics below to learn more about the updates in this release:

Base Program

- Fixed an issue where the system did not properly display sire sexed codes in sire names (for example, `GALLANT.S`), especially after milk loads.
- The nightly backups now include the Lactanet CANADA.DEF file used for electronic registrations (when available).
- When a new program version is used, we now include both the version date and the version number in the **Activity Log** message.
- The system now displays a warning message if you attempt to modify the disposal code, CAR code 6, which is generally used as a DIED disposal code.
- Detected and fixed item tables where Days Carrying Calf/Pregnant (DCC) was incorrectly defined. The tables were not showing the correct value for dead cow records.
- The system now displays a warning if you enter an ABORT date that is set before the latest PREG event date.
- Sires with a pooled semen NAAB code (such as 200PS1234) are now considered as beef sires for analysis purposes. While that is almost always the case, we are aware that a small number of PS bulls may be dairy, which may be addressed in a future release.
- We made an improvement to help prevent the program from locking up when you convert to the new activity log.

Integrations

- Nedap Now: Added support for configuring NEDAP Now alert items for PULSE.
- Waikato/AIC: Updated the integration name from AIC integration to Waikato Dairy HQ.
- ATA / other SSF SENDs: Previously, you had to add `\H` to the SSF SEND command (`SEND\DNLH`) in order to send health events to the DRPC. Now, this is the default for ATA herds and the `\H` is not required.
- Lactanet Traceability: Fixed an issue where MOVEIN events were missing the declaration date for the Simplitrace traceability system.
- Lactanet Traceability: Added a few requested extra items to the DairyTrace MOVEIN and MOVEOUT events.

CowCare

We removed the older `ALTER\7` protocol dialogs and now exclusively use the new CowCare dialogs.

PULSE

- When protocols were defined on PULSE, the protocol event number entered was not properly applied to the protocol added in DC305. We have resolved the issue.
- Fixed a problem that occurred during PULSE synchronization if you restored an accidentally SOLD animal and completed a XID ID change.
- Fixed a problem where PULSE did not properly clear the archive date item (ARDAT) if a SOLD or DIED event was removed in PULSE.

PCC

Added more vetcode details to the PCC refresh.

DairyComp 305 - Version 23.8

Release Date: August 29, 2023

DC305 version 23.8 contains the usual assortment of bug fixes, customer-requested enhancements, and integration improvements for DairyComp, PULSE, and our integration partners.

Expand the topics below to learn more about the updates in this release:

Base Program

- Previously, if you pressed the Ditto key (*) on a MOVE event, the system would incorrectly ask you to enter a pen. We have corrected the issue.
- Fixed an issue where the **Previous Lact** button wouldn't load properly on CowCards.
- Updated the USA sire file with August 2023 proofs.
- Added some information in **SETUP > Logon Settings** to explain why some buttons are disabled.
- Fixed the bottom axis scaling for graphs showing log SCC values.
- In FRESH, the freemartin status was not being determined properly in the case of triplets containing a male calf.
- Added a **SETUP** setting that allows you to auto-create the OTF (On the Fly) items at startup without having to enter a GUIDE command.
- Fixed how the system shows MAST protocols in EGRAPH reports.
- Added a new option (9) for Item Type #77 (BRED event info) that indicates whether the breeding is the result of an embryo transfer implant:
    - If the BRED event is the result of an embryo transfer, the item returns **E**.
    - If the BRED event is *not* the result of an embryo transfer, the item returns **N**.
- Changes to disposal codes (CAR codes) on the server are now properly sent to DCCLIENTs.
- Fixed problems with the **Find and Replace** feature in the **Held Commands** dialog box.
- We updated the list of stud codes that reference sexed semen, and we have set up automation to make the updates run more smoothly.
- The User Manager can now control the use of the semen inventory (SEMINV) module.
- Removed the VAS Community link button.

Integrations

- InterPulse: Integrated the import of the Average Milk Conductivity data from the `milking_data_YYYYMMDD_M#.csv` file into InterPulse/DairyComp 305.
- ScotEID: Fixed an issue where the ScotEID link was not sending births correctly.
- UK Dairies: Corrected a problem that occurred when the system showed sire breeds for UK herds.
- SmaXtec: Removed an unnecessary error message from the SmaxTec integration has been removed from activity logging.
- CowManager: Added support for the sort code item in CowManager.

CowCare

Fixed problems related to adding and deleting drugs from Inventory and merging multiple drugs together.

PULSE

We continue making improvements for various PULSE integrations issues regarding missing animals, as well as various problems with XID and restored FRESH events.

Web Updater and Backend

In the ABOUT nightly build message, we added an option to preview the changes that are available in the latest nightly build download.

Expand the topics below to learn more about new online help content!

Updated Help Topic: Manage Drug Usage Transactions in CowCare

We have updated the CowCare Drug Usage help topic to include CowCare's major upgrades! Use the new instructions to review, edit, and approve your drug usage transactions. Check out the new toolbar, keyboard shortcuts, and search and filter options that make managing your transactions easier than ever!

See **Manage Drug Usage Transactions** to learn more!

DairyComp 305 - Version 23.7

Release Date: August 3, 2023

DC305 version 23.7 is primarily a maintenance release, containing various bug fixes and improvements for the base program, integrations, CowCare, and PULSE.

Expand the topics below to learn more about the updates in this release:

Base Program

- Updated the German language file.
- We now prevent the deletion of Item Type #98 if it used as the Barn Name lookup item. If it is not, we highly recommend that you do not delete Item Type #98 because it could adversely affect DHI and partner integrations.
- Fixed an issue that occurred when the SETUP option to "auto-create a MOVE event when the premise changes" is enabled. Previously, if the option was enabled and you used the Ditto key (*) during event entry, DC305 would not let you enter a Protocol or Remark.
- Added performance improvements so that it's easier to enable Enhanced Activity Logs. To learn more about how you can quickly find and filter the entries you need using Enhanced Activity Logs, see **Activity Logs**.
- Alta stud code 699 is now correctly designated as sexed female. Previously, it was designated as sexed male.
- Due to a change in the sire data file we receive, some sire breeds were identified as “H” or “J”, while others were “HO” or “JE”. We addressed this inconsistency and continue to use single-character breed codes for the major dairy breeds.
- Added support for the ability to upload more than one PUTCOW file within the same minute.

Integrations

Lely: Fixed an error message that occurred during the Lely milk import.

CowCare

- Fixed an issue where the system would incorrectly write off drugs if you entered them on DCCLIENT.
- Resolved an issue where the drug editing window would not work properly in countries that use commas as numeric separators instead of decimals.
- Added support for Russian dairies to define premises, which can be used in CowCare.

PULSE

- Fixed an issue where, in some cases, PLCLIENT would generate an error that would block the synchronization between DC305 and PULSE.
- Resolved a problem where using an XID twice would stop the synchronization between DC305 and PULSE.

DairyComp 305 - Version 23.6

Release Date: June 27, 2023

DC305 version 23.6 contains an assortment of requested improvements, integrations updates, and minor bug fixes. We've also made updates to the latest version of CowCare. Many of our efforts in this release include internal changes in preparation for future features.

We've also made significant changes to the latest version of CowCare.

Our upcoming efforts include several requested improvements for PCC, PULSE reliability, Cowfile Version 9, and a variety of integrations updates.

Expand the topics below to learn more about the updates in this release:

Base Program

- Released updated May 2023 sire files for Italy and Canada.
- Resolved an issue where automatic incrementing of beef tags in FRESH was not working.
- Added a scrollbar to the ALTER : Dairy info dialog.
- Reduced the size of cowfile backups by removing the old Drugwatch files.
- We are now using an improved uploader for nightly backups. This reduces some of the issues we've seen with uploading backups for larger herds.

Integrations

- Delpro: Resolved an issue where EID formats for Morocco caused problems with the Delpro integration.
- Lactanet: Fixed a problem that occurred when sending MOVEOUT events to the Canadian SimpliTrace traceability system.
- Lely: Implemented a workaround for an issue that occurs when sending BNAME to the Lely T4C/Horizon integration. The integration now has an **Always send cow names** setting that is off by default. Use the `SETMILK\INSTALL` command to turn the **Always send cow names** setting on or off.
- When this setting is off, cow names are only sent to T4C/Horizon when the **Barn Names** setting is enabled in ALTER11. If the setting is on, cow names are always sent to T4C/Horizon. When a cow's name is present, that name is used. If a cow's name is missing, DC305 creates the name using a string representation of the cow's ID.
  ScotEID: Fixed a problem that occurred when entering destination holdings for the ScotEID traceability system.
- Zoetis: Resolved an issue where commands sent from the PULSE Zoetis genomics integration were being held if the commands were too long.
- AIC: Loading Milk Weights for AIC now ignores input records if "X" is at the start of the transponder code.

  > **NOTE:** The import generates an error file (AIC.ERR in the DC305 folder) that explains why other import files may have been skipped.

CowCare

- Added a notification window to alert that the old version of **CowCare** will be replaced on September 4, 2023.
- Fixed an editing drug settings issue.
- Implemented requested features for the new CowCare:
  - Added a setting to fail treatments if the DrugLot date is before the treatment.
  - Added a setting to filter only the DrugLots that have quantity.
  - CowCare now displays the number of treatments and total drug quantity on the bottom of the grid.
- Added the following date fields in Audit:
  - Date of treatment: Approval and reversal treatment
  - Delivery Date: Adding new inventory
  - Drug Loss Date: Adding new Drug Loss
  - Enter Date: Adjustment of Inventory
- Added the ability to report audit reports by either the date of a transaction or date of the record type.

PULSE

- Resolved an issue where backup programs interfered with PULSE synchronization and caused animals to be deleted.
- Fixed a problem where using an XID twice caused PULSE synchronization to stop.

PCC Pocket CowCard

Resolved an issue where item numbers over 255 were not properly sent to PCC entry lists.

DairyComp 305 - Version 23.5

Release Date: May 30, 2023

DC305 version 23.5 contains an assortment of requested improvements, integrations updates, and minor bug fixes.

We've completed efforts on fully supporting triplets in FRESH, including the associated items and reports. CowCare continues to improve with multiple requested changes from our partners and customers.

Our upcoming efforts include several requested improvements for PCC, PULSE reliability, Cowfile Version 9, and a variety of integrations updates.

Expand the topics below to learn more about the updates in this release:

Base Program

- Added support for triplets in the `EVENTS\3` Calf report.
- Fixed an error in the `CONNECT\A` Genomics (DNG) import that caused DC305 to hang.
- Reimplemented the Prostaglandin Analysis Option in BREDSUM that was removed in version 23.4. You can access the option from the BREDSUM menu or by using the `\PG` switch.
- Resolved an issue where the ALTER-Sires display grid was showing duplicate sires.
- Released sire files that contain the April 2023 sire proofs for Germany and the UK.
- Resolved an issue where the ALTER-Items display grid was showing a blank item at the top of the list.
- Corrected NAAB stud codes 602 - 605 to designate as sexed sires in ALTER8 and type 190 sire items.
- If a CLEANUP failure warning is displayed during DC305 startup, you can now run the `CLEANUP` command to clear the warning.
- Item types 73 and 76 now include the calf detail for a triplet calf to the item result. For example, three live triplet females added to the herd are displayed as "FKAFKAFKA."
  For proper triplet display, ensure `CONVERT\REM16` is used to extend the remark field length to 16 characters.

Integrations

- Fullwood: Some versions of Fullwood parlor integrations do not support DC305 sending lactation numbers. If an error is generated, add `\N` to the `MSEND` command to disable this value.
- ScotEID: Events that occurred "today" are now included, and a confirmation is displayed after the process is complete.
- SCR/Allflex: `SETMILK\INSTALL` no longer creates SCRTG. When HTTAG is created, it is now a type 32 item.
- Crystal-Fusion: By default, animals with a zero transponder are not sent to Crystal. This can result in duplicates if some heifers are assigned transponders. To avoid duplicates, add the `\H` switch to the Crystal-Fusion `MSEND` command to allow all animals, including heifers with zero transponders, to be sent to Crystal-Fusion. It may also be necessary to remove the `FOR` `TRANS>0` from the `MSEND` command.
- Lely: In a previous release, we implemented a change to resolve issues with barn names in the Lely `MSSEND` command. This caused issues for other Lely users, and the change was reverted in this release. We continue to work with the Lely team to properly send names to the T4C/Horizon system.

CowCare

- The new version of CowCare highlighted in our 23.2 release notes is now available! The update includes several major CowCare enhancements, including the option to run CowCare and DC305 concurrently, an updated dashboard with quick access to herd statistics, and significant speed/usability improvements. See **Install CowCare** to learn more about setting up the new CowCare.

  > **NOTE:** Existing DC305 customers can enable the new CowCare by going to **SETUP > Preferences** and clicking the **Turn on New** CowCare button.

- Resolved an issue where some CowCare protocol usage activities incorrectly displayed the event pen as 1023.
- If there was no entry of a delivery doc in the original CowCare, the new CowCare would not allow you to edit drug lots. You can now edit drug lots if the delivery doc is missing.
- Removed the ability to switch treatment from "No Treat" back to "Pending."
- Updated multi-purchase input to remove the user from the audit trail.
- Resolved an issue where the sort order on the treatment screen didn't work in the grid when a PC was set to US format and DairyComp was set to EU.
- Fixed an issue where CowCare prompted for a password multiple times.
- Resolved an issue where CowCare would not switch back to the activity screen when activity occurred in DC305.

Web Updater

Added error checks to resolve an issue where failed nightly backups caused program hangs.

PCC Pocket CowCard

Foreign language and English expert batch commands are now both recognized when sending data to PCC. For example, "`ENTER\Z`" and "`ENTRAR\Z`" are both recognized.

Expand the topic below to learn more about new online help content!

Updated CowCare Help Topics!

With the significant enhancements made to CowCare, we've updated many of our CowCare help topics! See **CowCare** to learn more!

DairyComp 305 - Version 23.4

Release Date: April 26, 2023

DC305 version 23.4 contains an assortment of improvements, requested integrations updates, and minor bug fixes.

We added more improvements to the FRESH command, including support for storing calf details for triplets. We've also added the option to group by Pen of the BRED event in BREDSUM. For use in certain genomic items, we expanded the type 42 item to allow a variable number of decimal places. Additionally, we are working on internal changes to the web update process as we continue to work on Cowfile 9.

Expand the topics below to learn more about the updates in this release:

Base Program

- Added the "By Event Pen" option to BREDSUM, which groups by the pen of the bred event. This option can be selected from the menu or by using the `\P` switch.
- Removed the "Heat Efficiency" and "Prostaglandin Analysis" options from BREDSUM.
- Added support for storing the calf details (live/dead, kept/sold, male/female) on triplet calves. This can be viewed in the FRESH event flyover hint. In this version, only the first two calves display. Version 23.5 will display triplet statuses as items for FRESH item types 73 and 76.
- Resolved an issue where entering `ESCAPE` from a picklist remark entry did not properly cancel the event entry. (for example, a `REM=RED;GREEN;BLUE` prompt)
- Fixed a problem where a badly defined item locked up ALTER2.
- Resolved an issue where `ECON\D` posted multiple unwanted alert messages when the entered herd date (EDAT) was unexpectedly missing.
- Expanded Item Type 42 to allow a selectable number of decimal places. The decimal places are specified in the second operator (Op2). The example below describes an Op2 value of 7654.321:
  - 4 Default: displays as $7654.32 (currency)
  - 1 One decimal: 7654.3
  - 2 Two decimals: 7654.32
  - 3 Three decimals: 7654.321
- MOVE events are now stored with both the original pen and the moved to pen. This removes the need to manipulate the remark field. Pens are available in the CowCard remark flyover hint and are used internally in various places including the CowCard page 6 pen movement chart.

Integrations

- Germany `SEND` : Added requested improvements for the German `SEND\V` to allow certain features to work across various LKVs.
- Germany `SEND` : Included "Causes of Impairing Performance" (CAR) codes in the KARIN.ADS export file.
- HeiferPro: Resolved a bug where DIED events were not properly recorded in the HeiferPro import.
- FTECH: Fixed the FTECH PC-DART import to support sire IDs longer than 9 characters.
- VMS: Added the registration number to DCMSEND.ISO (Item 5 Reg1 in Europe) to the VMS MSEND. This currently requires an INI setting.
- Lactanet: Continued improvements to SimpliTrace and DairyTrace.

CowCare

Updated the basis drug file with new drugs.

Expand the topic below to learn more about new online help content!

New help topics for Tru-Test SRS2 scanner users!

Does your dairy use Tru-Test SRS2 RFID scanners? Check out **Configure a Tru-Test SRS2 Scanner** for step-by-step configuration instructions! We've also included some instructions that show you how to delete sessions from the device. See **Delete a Tru-Test SRS2 Scanner Session** to learn how!

DairyComp 305 - Version 23.3

Release Date: April 3, 2023

DC305 version 23.3 contains improvements to CowCare, partner integrations, and a handful of enhancements and bug fixes.

We continue to make multiple improvements to the FRESH command. One highly requested improvement is the ability to maintain separate default ID and tag sequences for beef calves and crosses. On the ALTER > Dairy Info page, you can now define beef calf sequences. If the calf is determined to be a beef or beef cross, and

those fields contain values, then those fields are used for the default calf ID and tag entries. Future versions will include additional updates to the FRESH command.

Expand the topics below to learn more about the updates in this release:

Base Program

- Added the option to designate a default ID and Tag series for both male and female beef calves. This option is used when beef or crossbred beef calves are entered and is available in ALTER > Dairy Info.
- In ALTER > Items, DC305 now prevents a second item edit window from opening if one is already open.
- The ECON inventory report now properly accounts for user-defined sexed sires as well as NAAB coded sires when counting offspring.
- Resolved an issue where deleted cows were displayed when the `LIST \D` (dead) or `\B` switches were used.
- Fixed an issue that caused slow performance or system failures when searching for ET dams while entering FRESH in Version 9 cowfiles.
- Scheduled tasks were periodically hanging up after an error was produced in the connection to parlor or activity software. Connection hangups that exceed one hour are now terminated and logged.
- Fixed an issue where the Selective Dry Cow Therapy module was not installing correctly if the DRY event was not fully defined in ALTER - Events.
- Resolved an issue where the process for moving the maternity technician's name/number item to calves was not functioning.
- Fixed a problem with ABS sire stud codes 602 - 605 not properly displaying the sire info when the sire item was clicked.
- The default ID sequence on the server PC no longer updates if the **Use Client Calf ID Sequence** setting is enabled on the client.
- Extended the capability for prompted remark picklists. A value can now be prompted if the word `PROMPT` is in the picklist. In the following example, if **PROMPT** is selected, the user is prompted for a remark not on that prompted list:
  EC=42 REM=RED;GREEN;PROMPT;BLUE

  > **NOTE:** This feature is not supported in PCC or DC Mobile.

Integrations

- Fixed a bug where BULLPEN events were being inadvertently removed.
- The `MSEND` command to DairyPlan now only uses a BRED event (not a HEAT event) when sending insemination data.
- The system now sends the animal registration number (Item #5) through the BMOI integration.
- Requested changes have been made to the MNDHIA FTECH billing procedure.
- Fixed various issues when sending traceability data to the DairyTrace Canadian traceability system.
- Resolved an issue where the German GDTYP (cause of death) was not properly transferred from DCCLIENT and caused the DCCLIENT entry to be unavailable in EVENTS reports.

CowCare

- Fixed an issue where the premise number was not changed to the new premise when editing a pen number in CowCare usage entries.
- Resolved an issue with removing protocols after cows are SOLD or DIED.
- A new CowCare drug file for Italy has been deployed.

Expand the topic below to learn more about our updated online help content!

Updated Help Topic: Install DC305 on a New Computer

- We've updated the steps for installing DC305 on a new computer. This guide can be used if you are completing a fresh install or transferring DC305 data from your old computer to a new one. See **Install DC305 on a New Computer** to learn more!

DairyComp 305 - Version 23.2

Release Date: February 28, 2023

DC305 version 23.2 contains various bug fixes and improvements, with an emphasis on integrations updates. Improvements also continue to CowCare, the Cowfile 9

project, and various other enhancements and bug fixes.

Expand the topics below to learn more about the updates in this release:

Base Program

- The implemented a new "constructed ID" format for unknown genomic dam IDs. The new format was causing processing issues in DC305 and DRPCs. We implemented a fix to prevent against importing this new ID format.
- Fixed an issue where the `SHOW \QAT` average option was not averaging the type 150 series estimated weight items.
- Fixed an issue in `EVENTS` command reports where the report titles were not displaying correctly.
- Updated various language translations for DairyComp.
- Resolved an issue where the DCLAUNCH tool (used by DCCLIENT to receive updates from the server) was not working properly.

Integrations

- AgSource: Implemented the new DHI 43 "AgSource OnFarm" option for AgSource integrations. The new option fixes an issue where some of the extra DNL items, including 305DM and 305DF, were not updated in the `RECEIVE` command. The option is primarily used in the western US.
- Allflex/SCR: Resolved an issue where, in the most recent releases, DC305 was not properly calculating cows milked twice in the same milking.
- Boumatic: Corrected a format error that occurred when sending the lactation number through the BMOI integration.
- Boumatic: Added the `\R` switch to the `MSEND` command, allowing you to send lactation numbers, PREG/OPEN date, and PREG status to the Boumatic robots.
- Lactanet: Fixed an issue where records marked as "Deleted" in traceability logs were still being sent to the national database.
- Lely: Fixed an issue that occurred when sending barn names to T4C/Horizon. This issue may have been specific to Italian systems.
- ScotEID: Addressed and fixed an error where an incorrect URL was being directed to the ScotEID traceability system.
- Zoetis Genomics DND Import: Fixed a problem that occurred when loading the Zoetis Herdity items.
- USJersey: Added support for additional trait fields in the US Jersey DNI file import:
    - PTA Gestation Length
    - Health Trait Index
    - PTA Milk Fever
    - PTA Displaced Abomasum
    - PTA Ketosis
    - PTA Mastitis
    - PTA Metritis
    - PTA Retained Placenta

CowCare

- CowCare enhancements are coming your way soon! This release includes several major CowCare enhancements, including the option to run CowCare and DC305 concurrently, an updated dashboard with quick access to herd statistics, and significant speed/usability improvements. These new features will be field tested by a select group of users and will be publicly available in a future release.
- Fixed a problem that occurred when entering dose values on DCCLIENT systems.

PULSE

- Resolved an issue with duplicate animals with the same platform ID.

Web Updater/RS2 Replacement

- Removed an unwanted Cowfile 8 message in the updater. The error was displayed to a subset of users with older cowfiles.

Expand the topic below to learn more about our updated online help content!

Updated Help Topic: Activity Logs

- In the 22.8.0 release notes, we added a new Enhanced Activity Logs topic. We've since expanded the topic to cover all Activity Logging. See **Activity Logs** to learn more!

DairyComp 305 - Version 23.1

Release Date: February 1, 2023

DC305 version 23.1 contains various bug fixes and improvements, including changes for the base program, CowCare, and PULSE. Fixes and enhancements for integrations were a major focus in version 23.1.

Expand the topics below to learn more about the updates in this release:

Base Program

- An issue when using the `SIRES\9` command to import some sire items like PL and PTI has been resolved. The command not working properly was a result of a change made by NAAB with the sire file that DC305 imports.
- We fixed an issue where records sent to the new **Activity Logs** were being assigned incorrect categories.
- The December 2022 Sire files have been created for the UK and Germany.

Integrations

- Afikem: We added support for importing the "Non-specific" AFI alert in the `MILK\Q` alerts import. You can now run this import by adding `\M` to the existing `SEND` command. For example, `SEND\VM`.
- Germany `SEND`: We now allow alternate testing schemes for all LKs, not just Sachsen. You can do this by adding `\M` to the existing `SEND` command. For example, `SEND\VM`.
- Germany `SEND`: We resolved an issue where animal movements were not properly sent to HIT in Cowfile Version 9.
- Lactanet: We resolved an issue where CrossReference and TagRetired events were not always properly sent to DairyTrace.
- Lactanet: A new sire file containing the Decemeber 2022 sire proof data has been posted for Canada.
- UK `SEND`: An issue has been fixed where the `SEND` command used in the UK was generating errors when no premises were defined. Additionally, we clarified the message displayed when escaping the process.
- Zoetis `IMPORTF\D`: Zoetis Herdity support was added in version 22.12.0, but it was not properly loading some of the values. All assigned Herdity is now being imported correctly.

CowCare

- We fixed a problem where post-fresh withdrawal dates were being set for dry protocols with no withdrawals defined.
- We created and deployed a CowCare drug file for Czech Republic.

PULSE

- An issue has been resolved where some reports sent to PULSE for foreign languages, including German, did not properly display the "Totals" in the report footer and affected the PULSE widget display.
- A Cowfile error in the DC305 to PULSE synchronization process was interfering with PULSE uploads. This error was not previously displayed but has now been changed to a visible alert.

PCC Pocket CowCard

- We resolved an issue where some PCC users were receiving incorrect PCC license expiration messages.

Expand the topics below to learn more about new online help content!

New Help Topic: BREDSUM Reports

- BREDSUM in DC305 allows you to examine the effectiveness of your breeding program, including options for generating multiple reproductive statistics and measures. This help topic guides you through running a report, describes a report example, details the multiple command switches, and summarizes the syntax used with the command. See **BREDSUM Reports** to learn more!

New Help Topic: Parlor Reports

- The `PARLOR` command generates a report that summarizes daily milk activity. Data in the Parlor report allows you to find meters and machines that are not functioning properly, identify cows milked in the wrong pens, and monitor employee milking performance. This data can help guide any necessary changes to your milking processes. This help topic walks you through running a Parlor report and details the different sections of the report. See **Parlor Reports** to learn more!

New Help Topic: PCC Command

- Did you know you could create new commands for Pocket CowCard using DC305? See our new **Create a New PCC Command in DC305** FAQ topic to learn more!

DairyComp 305 - Version 22.12.0

Release Date: January 10, 2023

DC305 version 22.12.0 contains various bug fixes and improvements, including changes for the base program, integrations, and CowCare. We continued working on improvements for international traceability projects. Additionally, we have implemented multiple requests from new integration partners. We have also made improvements to our Cowfile 9 project, which continues to be field tested on multiple dairies.

Expand the topics below to learn more about the updates in this release:

Base Program

- We now properly calculate the dead/sold counts in INVOICE when events are logged after the DIED or SOLD event.
- An issue has been resolved where a cryptic and unnecessary message was displayed in the activity log at startup. The message was displayed in herds (primarily non-US) with a different setting for list separators including semicolons instead of commas in CSV lists, and commas instead of decimal points for numeric decimal values.
- By request, all vet codes (except CHECK and ABORT) are now cleared prior to the new vet list being generated. Previously, only the first 16 vet codes were cleared.
- We added a checkbox setting in ALTER Dairy Info to control whether the program should search archive files when looking for ETDAM information in FRESH. This was slowing down FRESH entry in large herds. By default, archive searching for ETDAM is now disabled unless the checkbox is set.
- We fixed an issue where you couldn't define a sire as "sexed" unless the unsexed variant also existed. This only affected dairies not using NAAB codes, primarily in Germany and Italy.
- We fixed an issue that occurred when viewing an EVENTS report and changing the date range using the **Options** button. Previously, the generated report would contain the correct data, but the report header would not properly display the newly selected date range.

Integrations

- Allflex/SCR: When the MSEND detects problems such as duplicate transponders, the conflicts are now set to an alert instead of being printed.
- UK/ScotEID: We resolved an issue where changes for ScotEID caused failures in the standard UK SEND.
  The ScotEID/ScotMoves project is still in field testing and has not been released.
- HeiferPro: When importing data from the HeiferPro import file, it will no longer overwrite existing EID (item #5) items if the import file has a blank value.
- HeiferPro: We've set the CAR to "6" for DIED events imported from HeiferPro.
- AFI: We have added the ability to load AFI activity files on DCCLIENT.
- Zoetis: DairyComp can now save the Teat Length (TL) genomic value from Zoetis import files.
- Zoetis: You can now import Herdity files into DairyComp from Zoetis DND files.
- Germany: Defining alternate test schemes was only available for Sachsen LVK. We now allow alternate schemes for any LKV if the **\M** switch is added to the **SEND** command.
- Lactanet: The ability to define sexed sire options in BRED for Lactanet herds has been removed.
- Boumatic: Lactation number, preg check date, and preg check results have been added to BMOI MSEND.
- Boumatic: DairyComp now allows spaces in the UNIV setup parameter for BMOI integration. For example, "C:\BM open interface\".

CowCare

- We fixed a problem that occurred when editing dosage on the usage screen.
- CowCare creates a CDOSE item used for manual dosages. If there was not room to create the item, DairyComp displayed a blocking dialog. This has been changed to a less obtrusive log entry.
- We added a setting that will display the actual withdrawal days instead of the calculated days on the inspector report. This is only used in some countries.
- You can now change the names of the drugs in CowCare. Additionally, you can now disable the use of certain drugs in protocols without completely removing the drugs.

PULSE

- An issue with Russian and Chinese herds where the calf widget was not displaying properly has been resolved.

DairyComp 305 - Version 22.11.0

Release Date: December 6, 2022

DC305 version 22.11.0 contains various bug fixes and improvements, including changes for the base program, integrations, and CowCare. This release also includes an improvement that speeds up commands such as BREDSUM by as much as 25%. Work continues on cowfile 9 which continues to be field tested on several dairies. Additionally, we made some internal changes for the PULSE integration, however these updates have no visible effect for the users.

Expand the topics below to learn more about the updates in this release:

Base Program

- We fixed the **SIRES** command options "Analysis - Sires of Cows" and "Analysis - Service Sires" to properly show the selected date ranges.
- The Background Refresh options for HTTP compressed and uncompressed in the Server config on DCClient are again operational. (The **Advanced** tab is available when using `SERVER\M`. For some time, attempting to use these options on a Refresh generated a "Server does not support HTTP refreshing" error and the refresh failed.) Cowfile version 9 users should use the uncompressed version because the cowfile is already compressed. Version 9 can use the compressed or uncompressed option. Overall, the background refresh option solves some issues, especially with large herds and those clients with slow or less reliable connections to their server.
- We were notified that stud 664 - Jetstream Genetics is a sexed female designation, and we made that correction.
- We released an updated Portuguese language file.
- We resolved an issue, possibly limited to Russian herds, where event tech teams were wrongly added to new events as they were created.
- We now allow spaces in the file names specified in FILEIN definition files used by `FILEIN\R`.

Integrations

- UK: We fixed the display for item type 134, which is only used in the UK for breed designations.
- SCR/Allflex: We removed the prompting for MTIM (milking times) in the SETMILK installer for SCR, which could cause a "File not Found" error in the milk load.
- SCR/Allflex: The commands that are created automatically with the SETMILK installer for SCR have been fixed.
- Scotland: We implemented the first release of the ScottEID traceability integration for Scotland.

CowCare

- We are now able to verify that all protocols are accounted for when opening CowCare.
- We fixed an issue where CowCare would accept a drug loss in the Inventory page, even if all fields were completed.
- We now allow drug doses between treatments in CowCare.
- We fixed an occasional error when entering purchases into the grid.

Web Updater / RS2 Replacement

- Installing Java is no longer necessary to update DairyComp 305 after this release and the previous 22.9.0 release.

PCC Pocket CowCard

- We added the PCC Device ID to the activity log for all entries made on PCC to allow you to see which particular device was used.

DairyComp 305 - Version 22.10.0

Release Date: November 8, 2022

DC305 version 22.10.0 contains various bug fixes and improvements, including changes for the base program, integrations, and CowCare. Additionally, we've implemented fixes for various milk integrations, imports of genetic indexes, several improvements for Lactanet, and some minor improvements for PULSE and PCC. We continue to field test the new, larger cowfile 9 on several dairies.

Expand the topics below to learn more about the updates in this release:

Base Program

- When one of the daily milk weights for today is zero, it now substitutes the weekly average for that milking in the calculation of the total, instead of the previous method of using yesterday's value for that milking.
- We fixed a problem that occurred when the system reloaded the new activity logs after switching between herds.
- We expanded the command entry size for ALTER-Commands. The system now displays the entire command when entering very long commands.
- We do not allow event remarks with certain characters, such as a backslash (\), because they cause problems with integrations, including PULSE. This same logic has now been added when editing event remarks.
- The use of Item Type #67 (lookup items) in vetlist constraints is now allowed.
- We added a checkbox in ALTER-Items to allow displaying either English or Foreign item names (use for Russian and Chinese herds).

Integrations

- Lactanet: The system now automatically updates the FRESH table to add the dam's ID (REGID) to the resulting calves' dam ID (DREG).
- Lactanet: Fixed an issue where the second calf in a set of twins was not getting the correct REGID.
- Lactanet: FTECH: Removed restrictions on changing the items on CowCard page 1.
- Boumatic: New NEDAP interface for Boumatic milk weights are in kilos. Also added a new `\C` switch to convert pounds, for example `MILK\T1CN`.
- ATA-Index: Resolved an issue where the Index import from ATA was incorrectly assigning some items.
- Lely: Fixed a problem with animals not marked as milkable due to Life Number issues.
- Lely: Fixed a problem with animals being removed from T4C/Horizon due to XID usage.

CowCare

- DC305 now saves the last entered delivery information as the default when adding new purchases.
- We fixed an error message that occurred at program startup, even when CowCare was not installed.
- We added a new "Inventory Quantity Report" that allows you to list inventory, either entirely, or by premise.
- We added the ability to fail approval if no drug delivery document was added.

PULSE

- We fixed a problem with duplicate Platform IDs that could have been caused by animal movements from a heifer grower.

Web Updater / RS2 Replacement

- Installing Java is no longer necessary to update DairyComp 305 after this release and the previous 22.9.0 release.

PCC Pocket CowCard

- We added the PCC Device ID to the activity log for all entries made on PCC to allow you to see which particular device was used.

DairyComp 305 - Version 22.9.0

Release Date: October 11, 2022

DC305 version 22.9.0 contains various bug fixes and improvements, including changes for the base program, integrations, and PULSE. However, in this release much of the focus was targeted towards fixes for CowCare and the Lely integration. Additional improvements were made to activity logging and web updater to support PCC updating.

A new RFID scanner can now be used to scan your animals with the mobile app. DC305 now supports the Allflex APR600 RFID scanner. For more information, see **APR600 Support Update**.

Expand the topics below to learn more about the updates in this release:

Base Program

- Activity data sent to DC305 through a PULSE integration, such as Allflex HeatTime, was being sent per-cow, which cluttered the activity logs. If the new **Activity Logs** feature is used, these are put into a new "Sensors" activity category, which can optionally be turned off in the configuration settings to keep them from the default activity log view.
- ALTER\8 Edit sire window now shows the sexed code status next to Breed and Stud if either of those entries define a sexed sire. For example, the breed AN determines that it's a Beef sire. This helps minimize confusion when a sexed sire does not have a suffix, and is necessary in countries where NAAB codes are not used.
- We fixed a small problem in COWVAL where the caption for Lactation numbers was duplicated (1, 2, 2 changed to 1, 2, 3).
- We resolved an issue where all of the switches were not properly saved when an EVENTS\R report was created and then re-run.
- 652NR Geno AI sires were improperly being coded as sexed male, but they are sexed female. This has been resolved.
- If you renamed the MILK item to something else, you were not able to rename it back to MILK. This has been resolved.
- We improved the "There is an update ready" notification at the bottom right corner of the screen.
- In some cases, creating and then canceling a new protocol entry was still creating the protocol. This has been resolved.

Integrations

- Lely: We fixed a problem with errors caused by duplicated Life Numbers.
- Lely: We fixed a problem updating the complete T4C records for new animals added.
- Lely: We resolved an issue caused by blank barn names.
- Any integration setup dialogs that create items will now log those item additions in the activity log.
- For the new **Activity Logs**, all integrations (milk loads, milk sends, activity loads) are logged along with the vendor name as Integrations.

CowCare

- We fixed a problem where custom heifer dosages were not reporting correctly.
- We resolved an issue where delivery document IDs were not being assigned in the correct order.
- If the checkbox that says “don’t allow approvals with negative quantities” is checked, the system will now properly stop the approval of any treatments when it gets to zero.
- Some treatments were missing from CowCare, which was caused by the user waiting too long (sometimes months) to update CowCare. With this change, CowCare is automatically updated on a daily basis.
- When the CowCare database is resotred from backup, duplicate treatments are no longer being added.

PULSE

- DairyComp and PULSE handled blank sires differently (zero vs UNKNOWN) in some cases. This is now fixed.
- We resolved an issue where deleting a FRESH event in PULSE was causing an error in DairyComp.

Web Updater / RS2 Replacement

- We added support for PCC updating in our new web updater, which makes us close to completing our switch away from Java. The changes are in this version, but won't take effect until a second web update is run.

PCC Pocket CowCard

- The activity log now shows EID values entered from PCC Add Batch commands.

New! DairyComp 305 now supports the Allflex APR600 RFID scanner

In addition to the Allflex AWR300 RFID scanner, DairyComp 305 now also supports the Allflex APR600 RFID scanner scanner, allowing you to scan animals with the mobile app. The APR600 can be paired to your Pocket CowCard for Android device.

See **Connect an Allflex Scanner** to learn more!

DairyComp 305 - Version 22.8.0

Release Date: September 7, 2022

DC305 version 22.8.0 contains various bug fixes and improvements, including changes for the base program, integrations, and PULSE.

One long-awaited improvement is the first release of our enhanced Activity Log. We have made it easier for you to filter and find the entries you need! When enabled, activities and entries are grouped into categories. You can then view activities and entries by category or date, or hide categories from the log. For more information, see **New Help Topic: Enhanced Activity Logs**.

Most of the work we have done this release cycle has been in the development of our new cowfile 9 version. This contains more space for cow items, and events, and we have expanded the allowable number of event codes, protocols, commands, and most of the ALTER options. Progress has been good, and we've made many improvements based on our early feedback. We are starting to release it for early preview to a subset of our customers.

Expand the topics below to learn more about the updates in this release:

Base Program

- We were receiving reports of technician teams being prompted when the event or protocol did not define a team. This issue has now been fixed.
- The updated sire proofs were released as part of version 22.8.
- We fixed a problem where in some cases the event TECH entered on the client was not properly sent to the server.
- We resolved an issue where running the GETCOW**\I** command did not find the DNC file that was placed in the DC305 folder, which required manually changing the default folder selecting the file.
- A bug as been fixed where MONITOR parameters using @MTR (since last MONITOR was run) were not seeing more than four constraints. MONITOR parameters now recognize all six constraints if present.
- A printer is required to be defined in order to run DairyComp. We have improved the detection of a missing printer, which was causing the program to stall if it was unable to detect a printer.
- Running certain batch files within DC305 would periodically write an error message to the activity log. This was determined to be a false error message and has been removed.
- We fixed a bug where the ALTER7 : Protocols command was allowing mapping of the same event to more than one protocol event, or the same item to more than one protocol item.

Integrations

- We added the Jersey Neuropathy (JNS) trait to the Zoetis DNG import.
- The SCIMM (Semex Calf Immunity Index Value) trait has been added to the Semex DNG import.
- We added a new DHI type for AgSource Onfarm herds. This is primarily for FTECH and used only for AgSource herds using . This should not be used for typical AgSource herds using DC305 that are tested with FTECH.
- We fixed a problem where the FTECH import for PC-DART herds was not properly supporting 2-character breed codes.
- The Stature (GTSA) trait has been added to the Zoetis DNG import.
- We fixed various problems with foreign language event names in the DairyMaster integration.
- We improved the rounding of dairy milk weights imported from the SCR integration.

PULSE

- The number of days for saving past worklists sent to PULSE has been reduced. This will free up disk space on customer computers.
- A bug has been fixed where some combinations of XID were not being sent properly to PULSE which blocked further updates.

Web Updater / RS2 Replacement

- We've made various improvements to both the web updater and the RS3 ZOF downloader. At this time, we are unable to go completely Java free. However, the DC305 team continues its work with the PCC team on downloading the necessary updates, which is being developed for DC305 version 22.9.

New Help Topic: Enhanced Activity Logs

When you enable the enhanced Activity Log, you can quickly filter and find the entries you need! Activities and entries are grouped into categories, which can be selected by category, by date, or made visible or invisible in the log. To enable this feature, enter the SETUP command in the command line and go to the **Preferences** tab. Toggle on **Enhanced Logging**.

When you first enable the new Activity Logs, it might take a few minutes to create the Activity Log database using your existing log files. The system will continue to update the default text log files even if you enable or disabled in the enhanced Activity Log.

Additionally, we're saving more details for each entry. In upcoming releases, we plan to add more activity logging features. See **Activity Logs** to learn more!

DairyComp 305 - Version 22.7.0

Release Date: August 4, 2022

DairyComp version 22.7.0 contains various bug fixes and improvements, including changes for CowCare, Integrations, and the base program. This release also includes ongoing fixes and improvements for PULSE integrations and new update process. Several bug fixes and requests from our customers and partners have also been addressed.

As we proceed, we continue to work on the upcoming Cowfile Version 9 with expanded data storage, as well as an enhanced version of activity logging, which will be available in a future release.

Expand the topics below to learn more about the updates in this release:

Base Program

- We fixed a problem where the system would not properly show "B" for beef sires that didn't have a NAAB code. This change fixes type #190 items, as well as **BREDSUM*** (BREDSUM by sexed code).
- We removed the **ECON Income over feed** cost option, which was not being used.
- We deployed a new German language translation file supplied by the German support team.
- Due to an error at the Council of Dairy Cattle Breeding (CDCB), a new USA sire file was received. The correct version has been deployed to customers.
- By request, we added a button in ALTER8 : Sire Table so you can directly go to the Semen Inventory.
- Some customers wanted to remove the ability to enter FRESH if a cow is not pregnant. To achieve this, enter SETUP in the command line, go to the **Parameters** tab, and set the **Minimum DCC to allow FRESH** option to a nonzero value. Note that this blocks FRESH, unlike the existing DC305 option, which is just a warning that can be bypassed.
- When the **EVENTS\2** command was used to list events, the system only displayed the tech for BRED events. The system now properly shows the event tech used for each event listed.
- Resolved an "Out of bounds Exception" error in DC305 that occurred when the system ran the **SAVE** or **STATUS** command for some herds. This was traced to a corrupted temporary file used by CowCare.

CowCare

- The system was not setting the Milk withdrawal date (MKDAT) correctly if you used **DRY** protocols in CowCare and if the protocol was set to add a number of days to the fresh date when the FRESH event occurred. This has been corrected.
- The CowCare module was inadvertently being activated whenever the ALTER7 : Protocols configuration was entered, even if you didn't want to use CowCare. This has been fixed, and this will also clear any accidental CowCare databases that may have been created.

Integrations

- You can now set the time on your reports to show either 24-hour time (14:30) or 12-hour time (2:30 PM). To change the setting, enter **SETUP** in the command line, go to the **Printer** tab, and update the **Show Time On Reports** option.
- We fixed a problem for some dairies where loading Allflex Heatime data would cause the system to lock up DC305 with an "out of bounds" error message.
- We fixed a problem with the Canadian Traceability process on DCCLIENT. If births were entered on the client and a different pen/premise was used, the system did not record premise changes properly on the server Traceability system.
- We fixed a mistake where the system would send the wrong ear tag when users entered the German version of the **SEND** command.
- We fixed a problem with the UK British Cattle Movement Service (BCMS)'s **SEND** command where it incorrectly identified Wagyu crossbred calves.

Web Updater

- The system now updates language (DCLANG) files whenever translations are modified, even if the files are updated between DC305 releases.
- There have been several improvements in the web updater to increase reliability and provide an improved experience.

Pocket CowCard

- We now log entries made on PCC separate from typical transactions. They are no longer defined as "V" transactions. Now, PCC entries are tagged with an "M" for "Mobile" in the default activity logs.

Expand the topics below to learn more about new online help content!

New Support FAQ: Check the DC305 and PULSE Connection

Did you know you can check the connection status between DairyComp and PULSE directly in DC305? Check out our new FAQ to learn how to check the connection and solve your connectivity issues! See **Check the DC305 and PULSE Connection** to learn more!

New Help Topic: Adjust the Hold Date (HLDAT)

Need to adjust the hold date (HLDAT) for an animal? If you want to schedule a vet check for your animal earlier or later than the current hold date allows, see **Adjust the Hold Date (HLDAT)** to learn how!

New Help Topic: Determine Why DNB Cows are on a Vet List

Is there a Do Not Breed (DNB) animal on your vet list? See **Why are DNB cows on my vet list?** to learn what might have happened and how to fix it.

DairyComp 305 - Version 22.6.0

Release Date: July 6, 2022

DairyComp version 22.6.0 contains various bug fixes and improvements, including changes for PULSE, CowCare, Integrations, and the base program. This release also includes ongoing fixes and improvements for PULSE integrations and new update process. Several bug fixes and requests from our customers and partners have also been addressed.

Improvements continue for expanding storage for cowfiles, enhancing activity logging, and improving the web updater, as well as migrating users to the non-Java version.

As we proceed, we continue to work on the upcoming Cowfile Version 9 with expanded data storage, as well as an enhanced version of activity logging, which will be available in a future release.

One long-awaited improvement in this version is enhanced support for sexed sires for customers who do not use NAAB codes. See **Enhanced Support for Sexed Sires**

below for details!

Expand the topics below to learn more about the updates in this release:

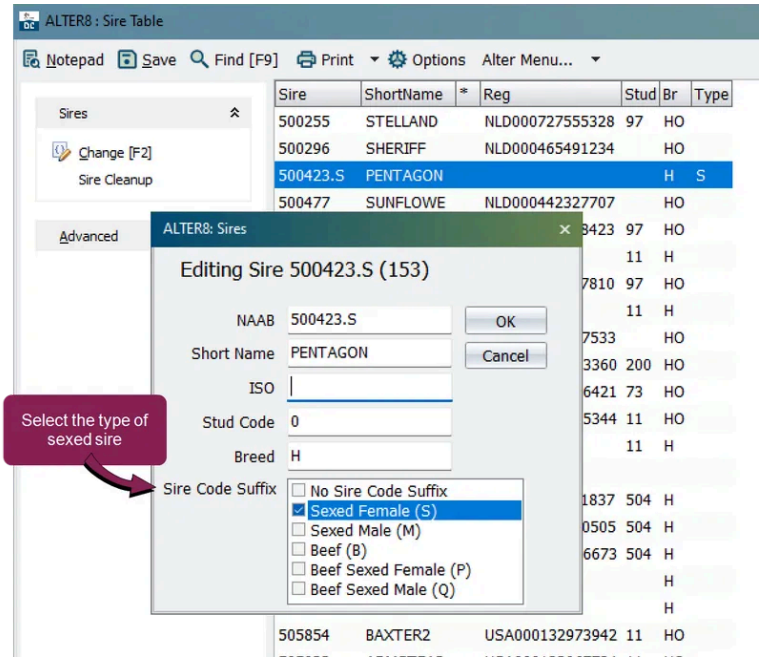
Enhanced Support for Sexed Sires

Customers who do not use NAAB codes can now use the enhanced Sexed Sire feature in DC305 to designate a sire as sexed female, sexed male, and sexed and unsexed beef sires.

For US customers, sexed sires are handled well already, as they are based on NAAB stud codes in the sire names. But NAAB codes are not typically available for our international customers, especially in Europe. To meet the needs of customers who do not use NAAB codes, DC305 now allows you to designate a sire as sexed female, sexed male, and sexed and unsexed beef sires.

The simplest method is to add **.S** to any sire name to designate it as sexed when entering it into DC305. For example **12345** is not sexed and **12345.S** is sexed.The command for the sexed sire would look like this:**ENTER SID=12345.S**

The more advanced method is to edit the sire in the ALTER**8 Sires** Configuration window. To designate a sire as sexed without relying on a NAAB code, you can use the new **Sire Code Suffix** field on the sire record to select the various sire sexed types (a straw, specifically).



See **Using Sexed Sires** for a more thorough explanation of this new feature, as well as detailed instructions!

Base Program

- The MONITOR report would display some values as negative numbers. This overflow was seen in very large herds, and has been fixed.
- The system was not showing the correct calculation for item type #110 (item divided by item). This is now resolved, however, some divided items may contain values that the system won't include in the report. For example, milk weights stored to half unit values.
- Updates for DCSquirrel.exe have been added to all download channels, not just PULSE channels.
- VAS monitors the NAAB stud code list to keep track of any changes, including sexed sire stud codes. This update contains some of these additions, including added support for **designating any bull as sexed semen without using NAAB codes)**.
- The **XID** command was not accepting IDs that had more than 9 digits. This has been fixed.
- An issue where the SIRES importer would occasionally import sire items over critical system items, such as BDAT or FDAT, has been fixed.

CowCare

- Previously, users could only collect treatments data from the previous 15 days. Now, users can go back 180 days to collect treatments data.
- Added the technician name to the Inspector report.
- Removed some prompts that were not applicable in certain countries.
- In some CowCare instances, the system was not showing dates in the user-selected format i.e. in Day/Month/Year or Month/Day/Year format. This is now fixed.

Integrations

- Fixed a problem in the HeiferPro import where some health events were incorrectly imported as PREG.
- When only using milk import files, it can be dificult to quickly determine whether a parlor is rotary or not. To force the setting, we added the **\R** switch support to the **PARLOR** command:
  - \R0 means that the parlor is not a rotary.
  - \R1 means that the parlor is a rotary.
- Fixed an incorrect caption on the setup screen for the SCR integration.
- In some countries (including Morocco), the system now sends fully numeric ISO codes to Delpro.
- Completed more work for the Lactanet integration.

PULSE

- Fixed some errors found in the PPK platform ID import from PULSE, in response to changes in the PULSE API.
- Fixed some issues for cows missing the correct PULSE IDs.

Expand the topics below to learn more about new online help content!

New Help Topic! Using Sexed Sires

Customers who do not use NAAB codes can now use the enhanced Sexed Sire feature in DC305 to designate a sire as sexed female, sexed male, and sexed and unsexed beef sires.

See our new **Using Sexed Sires** topic for help designating any sire as sexed, without relying on NAAB codes!

DairyComp 305 - Version 22.5.0

Release Date: May 31, 2022

DairyComp version 22.5.0 contains various bug fixes and improvements, including changes for PULSE, CowCare, Integrations, Pocket CowCard, and various web updater

fixes. We also addressed several bug fixes and requests from our customers and partners.

We continue to work on expanding storage for cowfiles, enhancing activity logging, and improving the web updater, as well as migrating users to the non-Java version.

Expand the topics below to learn more about the updates in this release:

PULSE

- Fixed a Squirrel error message that was caused by a partial, incomplete PULSE install. Protections were put in place to resolve this error.
- Fixed an issue in initial export to PULSE that had a corrupted archive file.
- Continuous improvements are being made to integrations between PULSE and DairyComp for entries made on either system.

CowCare

- Fixed an issue where protocols using the same drug multiple times were not calculating correctly.
- The order of assigning the Delivery Doc to the treatment has been changed.
- CowCare backup files are now included in the daily backup and SAVE\F commands.
- Fixed an issue where the inventory was not displaying properly for some customers.
- CowCare DRY protocols were updated to set correct withdrawal dates to begin when the cow freshens.

Integrations

- The Squirrel companion program, which is used for integrations including Lely and Smartbow, is now automatically updated on client computers when DC305 is updated.
- The UK SEND file now includes the dam's eartag number.

Pocket CowCard

- Applied a hotfix to version 22.4 to address a missing prompt for ID in some PCC worklists.
- You can now use MP3 and WAV files for sounds in PCC. Support for these files is also offered in DC305.

Web Updater

- Fixed failing pConnect nightly backups to PULSE on older, Windows 7 computers.
- Web Updater includes several notable enhancements, including an improved ability to determine if updates are available, and better handling of language and sire files.

General Enhancements and Changes

- Fixed an issue with event entry using the numeric keypad.
- Posted an updated German sire file with recent changes to the DHV numbers.
- Fixed an error in the Russian Translation file.
- Fixed a problem where the Import EIDs manager option was not reading cow IDs exceeding eight digits.
- SIRES options 7 and 8 for analysis now display correct date ranges.
- In the last release, we updated the formula for ECM. In this release, we also added the updated formula to be used in the type 129 projected ECM and relative value ECM items.
- The FTECH SEND summary report now displays IDs longer than six digits.

Expand the topics below to learn more about new online help content!

New Help Topic: Milk Production Over Time!

Need to see your milk production over a period of time? Our new help topic illustrates how DairyComp displays charts and graphs to help you better understand and visualize milk production over time. See **Milk Production Over Time** to learn more!

DairyComp 305 - Version 22.4.0

Release Date: April 27, 2022

DairyComp version 22.4.0 contains various bug fixes and improvements, including changes for PULSE, CowCare, Integrations and various web updater fixes. We also addressed several bug fixes and requests from our customers and partners.

We continue to work on expanding storage for cowfiles, enhancing activity logging, and improving the web updater, as well as migrating users to the non-Java version.

Expand the topics below to learn more about the updates in this release:

PULSE

- We continue to work on issues related to sending complete previous lactation records to PULSE.
- Fixed an issue where the system would display a "Duplicates not allowed" message and block some widget from generating.

CowCare

- We fixed an issue with CowCare reports opening without immediately displaying the animal list. The system now displays the animal list as expected.
- When starting a shipment in CowCare, the default shipment value is now set to zero. In rare cases, CowCare would freeze when CowCare shipments were saved without a quantity. This only occurred when CowCare was force closed during shipment entry.
- Added the ability to adjust the standard drug dosage when entering a protocol. You can now set up the protocol to allow these changes, and the dosage details are then transferred to CowCare.

Integrations

- Lely: Solved an ongoing problem that affected sending PREG/OPEN statuses and PREV events to the Lely T4C system.
- Added support for Canada's secondary traceability system called SimpliTrace. It is primarily used for Western Canadian dairies and is similar to the existing DairyTrace. Our Lactanet partners are now reviewing our support development.
- Added a requested field to the German SEND command (SEND\W and SEND\DW) that indicates whether the calving was a result of an embryo transfer breeding.

Web Updater

- We started updating the back-end destination of the nightly backups. You may notice background messages referring to this change, but the update does not affect your backups.
- Implemented some minor display changes in the web updater process.

General Enhancements and Changes

- Updated Stud 516 to a sexed female stud code, in compliance with the Global Ag Alliance's classification. This affects the type 190 sire sex item (%190.SID.7) for this stud, which is now "S."
- Fixed an issue where the Energy Correct Milk item (ECM, Type 85 with Op 1 of 2) was calculated improperly when either of the component values (PctF and PctP) were set to zero.
  While investigating that, we discovered and implemented newer formula for ECM. The new formula will have a small but statistically significant change on the values reported.
  The formula is:
  `ECM = (0.327 * MilkPounds) + (12.95 * FatPounds) + (7.2 * ProteinPounds)`
- When you update your program, the system will automatically download updated sire files (DCSIRES.xxx) for the USA, Germany, UK, and Italy. The new files include information from the newly-released April 2022 sire proofs.
- Removed the VAS Community button for international users.
- You can now configure client computers to perform automatic backups. To enable this feature, enter the SERVER\M command or go the Advanced Settings tab and check the new “Enable Client Backup Task” box.
- Addressed an issue where the system would enable User Manager after it was disabled. Now, when you disable User Manager, a system message prompts you to keep the existing user list while still setting it to disabled.
- Fixed an issue where the system would resend an event's previous changes. This occurred on DCClient if the user started entering edits for an event, but closed the edits before completing them.
- Fixed the CowCard Page5 PREVLACT display for Russian herds.
- Fixed the Type 122 Week 1 milk average item. Previously, it would incorrectly return the second week's value.
- Log searches in the Activity Log are now sorted from oldest to newest. A previous Activity Log update had changed how log searches were sorted.
- Previously, we created ETBRED events to indicate when an embryo transfer breeding occurred. In a few herds, the system identified some breedings as

ETBRED even though no embryo transfer had occurred. We fixed one cause, which occurred whenever an ET flag was incorrectly set during some breedings.

- This can also occur if users misuse the ETDAM item. Do not use the ETDAM item for anything other than an actual donor ID for embryo transfer breeding. It is a system item and DairyComp uses it for that specific purpose.
  In the previous release, we improved the activity log to distinguish between external entries made in DairyComp, PULSE, or iTrans. In this release, the Pocket CowCard label is now available. The entries now include the following labels:
  - D: DairyComp
  - P: PULSE
  - N: iTrans
  - M: Pocket CowCard

Expand the topics below to learn more about new online help content!

New Support FAQ: Update PCC on your Android Device

The PCC app is updated on a regular basis and often includes key enhancements and bug fixes that improve app performance and usability. Most versions of PCC have an in-app updater that automatically installs the latest version of PCC onto your device. If your version of PCC does not automatically download the latest version of PCC, manually download the latest version of PCC. **Before initiating any update, ensure all pending transactions have been delivered to** DairyComp. Please note that your PCC device must be charged to at least 20% before you can download the latest version of PCC.

See **Manually Update PCC on your Android Device** to learn more!

New Support FAQ: Backup PCC Transactions on Android 11+ Devices

PCC offers the option to backup transactions, which comes in handy while your device is offline and unable to send transactions to DC305. Use this feature if your connection is unstable or if your device is failing or crashing for unknown reasons.

See **Backup PCC Transactions** to learn more!

DairyComp 305 - Version 22.3.0

Release Date: March 30, 2022

DairyComp version 22.3.0 contains various bug fixes and improvements, including changes for PULSE, CowCare, Lely, and various web updater fixes. We also addressed several bug fixes and additions requested by customers and partner support.

We continue to work on expanding storage for cowfiles, enhancing activity logging, and improving the web updater, as well as migrating users to the non-Java version.

Expand the topics below to learn more about the updates in this release:

PULSE

- Corrected an error that occurred when a user removed a FRESH event from an animal. An error would display in an animal's data on PULSE if the system could not restore the animal's previous records.
- Added standard item numbers and updated the user interface for SIR3 and SIR4 to support the PULSE genetic planning tool.
- Added the PLATFORM command, which allows users to access all of the DC305-PULSE item links.
- Redirected ZOF uploads from iLoop to PULSE for 10% of all herds (herdcodes that end in "9"). The ZOFs will still go to iLoop at this time. This update may slow nightly backups.
- Improved the PLCLIENT sires tables to ensure that sire names are updated in both DC305 and PULSE if sires are renamed in DC305.
- Fixed an issue where DC305 would not work correctly when a file containing PULSE IDs included values from bad EIDs.
- Corrected an issue where some users would receive an access violation error when they opened PLCLIENT.

CowCare

- Changed the system's default settings so users can't add treatments without drugs or treatments without protocols. You can enable the option to add treatments without drugs or protocols in **CowCare** > **Settings** > **Inventory**.

Integrations

- Lely: Corrected the way we send ABORTS data to Lely's T4C management system to match the way Lely's T4C management system maintains pregnancy statuses.

Web Updater

- Fixed a problem where switching between different cowfiles caused an incorrect expiration message.
- Improved DCUpdater so the system now skips some Curtis steps if Curtis is unresponsive. The system will alert the user if any steps were missed. However, issues must be reported internally to DCUpdater because FetchUpdates will not make any calls to Curtis when this occurs.
- Corrected an issue where the background web downloader window would display in front of the DC305 window.
- Properly handled EULA documents in the web updater.

General Enhancements and Changes

- Fixed an issue where the Selective Dry Cow Therapy install module was failing for Russian herds.
- The User Manager can now assign permissions to specific ALTER features, not just the ALTER command overall. These permissions can be found in a new tab in the User Manager feature.
- Previously, when users created a FRESH event by entering a command on DCCLIENT, the system would not recognize any portion of the command that preceded the FRESH event. Now, the system will support all parts of the FRESH command.
  For example, if a user wrote `COD1=99 EC=1 EDAY`, the system would not recognize the item (`COD1=99`) because it precedes the FRESH event (`EC=1`). Now, the system will properly recognize the item even if it precedes the FRESH event.
- Corrected an issue where in some cases, the iTrans download would cause duplicate events when the event already existed in DC305, but had a protocol.
- Fixed an error where users with very large herds would see an Out of Memory error if they attempted to search the activity logs.
- The system now records changes made using the ALTER9 command (event codes) in the activity log.
- Corrected a translation file error that occurred when users tried to add new items to the TestDays tab on an animal's CowCard.
- Improved the Export to Excel option. Now, users can check the Improve Excel Export box in SETUP to speed up the process.
- Fixed an issue where STF LOAD incorrectly read STF "TOBULL" events as BRED events. This issue affected subsequent PREG events.

DairyComp 305 - Version 22.2.0

Release Date: March 9, 2022

DairyComp version 22.2.0 contains various bug fixes and improvements, including changes for PULSE, CowCare, Lely, and various web updater fixes. We also addressed over a dozen changes and additions requested by customers and partner support.

Expand the topics below to learn more about the updates in this release:

PULSE

- Fixed a problem related to sending Change IDs (XID).
- Resolved an issue where duplicate FRESH events caused unwanted calf ID changes.
- Fixed a problem where PLCLIENT failed to send updates to PULSE when a user logged on to different cowfiles.
- Added activity log labels for incoming PULSE commands. Expand **Activity Logging** to learn more.
- Added new safeguards to prevent accidental animal deletion in PULSE.
- Added a PULSE standard item STOC (semen type of choice) to support the PULSE genetic planning tool.

CowCare

- Added the option to enter and define a drug dosage for protocols when entering protocol events.

Activity Logging

- Improved the activity log to distinguish between external entries made in DairyComp, PULSE, iTrans, or PCC. Previously, the entries were labeled with a “V”. The entries now include the following labels:
  - D: DairyComp
  - P: PULSE
  - N: iTrans
  - M: PCC

    > **NOTE:** This feature is not available in PCC, but it will be available in the next PCC update.

- So, for example, if you see a breeding event in the activity log labeled as “P”, it was entered on PULSE.

Integrations

- InterPuls: Resolved an issue where blank values were occasionally included in the Interpuls milk import.
- Lely: Fixed problems related to duplicate Life Numbers.
- Lely: Updated the way we send ABORT data to better match the way Lely's system resolves OPEN and PREG statuses.

PCC

- Created a simple method to reimport lost PCC transactions. Contact **VAS Support** for more information.

General Enhancements and Changes

- Added a warning when a SOLD or DIED event is entered before the animal's last event.
- Fixed a problem where HeiferPro PREG events were incorrectly labeled as PREV events.
- The loading time for the SCR and SmartBow sensor data pages has been significantly improved.
- Resolved an issue where the CHECK vet code would not clear when a CHECK event was erased.
- Fixed a problem related to the tech remap function.
- Added functionality to support some DHI SENDs on DC305 computers.
- Fixed a problem in DCLAUNCH where the DC305 client did not always update correctly after the DC305 server was updated.
- Fixed an expiration date problem that occasionally affected subscriptions that were temporarily extended.

Expand the topic below to learn more about the new online help content!

New Support FAQ: Connect an AWR300 Scanner

Need to connect an Allflex AWR300 RFID scanner to an Android device for use with the Pocket CowCard app? See **Connect an Allflex Scanner** to learn more!

DairyComp 305 - Version 22.1.0

Release Date: February 4, 2022

DairyComp version 22.1.0 contains various bug fixes and improvements, including changes for PULSE, CowCare, Lely, and various DHI-related fixes. We also addressed over a dozen changes and additions requested by customers and partner support.

Expand the topics below to learn more about the updates in this release:

CowCare

- Fixed a problem in which protocols lost drug mapping after remapping using the `ALTER\7S` command.
- Fixed a problem with deleting unused inventory records.
- Made another significant improvement in CowCare speed.

PULSE

- Fixed a problem where the wrong animals were showing as deleted when a DNC file of imported animals was imported twice.
- The PULSE Dairy ID is now displayed for reference purposes in the PLATFORM command window.
- Various other work was performed to improve data integrity between PULSE and DairyComp.

Integrations

- Beco: Fixed flow rates in Beco parlor reports.
- Made improvement in sending FRESH information to Lely/T4C.
- Standardized the international ID format sent to Lely/T4C.
- Nedap: Fixed heat time and activity items to make gap items work properly.
- ParlorBoss: EID items are now always standardized to numeric.
- ScotMoves: Fixed various issues with the ScotMoves API.
- Lactanet: Fixed regression in Lactanet Traceability.
- Lactanet: Fixed incorrect calf sex for stillborn calves.

Web Updater

- DC Squirrel now properly closes before running the Java updater on non-Platform enabled herds.
- The new (non-Java) updater is the default updater from now on (except for hardlocked versions).

General Enhancements and Changes

- Allow `SETDAY` as a command that can be controlled with the User Manager.
- Handle duplicate BRED problems caused by the user of sire short names in iTrans.
- Fixed Selective Dry Cow install module for Russian herds.
- Fixed German DGTYP item duplication when foreign DCITEMS file is in use.
- Log all changes made in the `ALTER\4` Pens window.
- Fixed previous problem in which FTECH caused an error message upon exiting.
- Allow technician event teams to be deleted through right-clicking on the teams selector box.
- Properly export decimal values in the Save to Excel option.
- Properly save the PEN in ABORT events that start a new lactation.

Expand these additional topics to browse new online help content!

New Support FAQ: Troubleshoot Start-Up Errors

Have you ever had a problem launching DairyComp? Check out our new FAQ for answers to the most common DairyComp start-up problems! See **Troubleshoot Start-Up Errors** to learn more!

DairyComp 305 - Version 21.12.0

Release Date: January 4, 2022

This update is primarily a maintenance release. We've continued to update CowCare and various vendor integrations and improve PULSE integration. We've also made miscellaneous improvements and bug fixes in the main program.

We are completing the final work on the new non-Java Web Updater, which is supported in this version (but is not yet the default).

In the upcoming quarter we will continue to improve the Lely integration, and Cowfile 9 will undergo a thorough beta testing phase. We will also continue to handle various requests from customers and support.

Expand the topics below to learn more about the updates in this release:

CowCare

- CowCare properly removes usages after a SOLD or DIED event.

PCC

- Added support for Scanner Device ID item.

PULSE

- Fixed deletion of FRESH event when ETDAM is not found in cowfile.
- Fixed memory overflow of platform ID import in large herds.

Integrations

- Beco: Added support for Taurus format for Beco.
- Nedap: Revised the pregnancy check result value for NEDAP based on NEDAP requests.
- Lactanet: Fixed Calving Ease format with Lactanet XML Export.
- Lactanet: Ensured REGID item is updated properly in RECEIVE.
- Added support for Scotland ScotMoves Traceability Export.
- Avoid potential hanging if Delpro milk import fails.
- Fixed correcting milk weights (including 4x) in herds when some pens use robot milkers.

Genomics

- DND Import—Import Cholesterol Deficiency (GHCD) values.
- Import Jersey JNSC values.

General Enhancements and Changes

- Support more items for "EVENTS\3 BY..." (e.g., calf technicians).
- Fixed moving technicians between event and breeding teams when they have long names.
- Fixed display cutoff issue in EVENTS LAME\50.
- Reset RPRO if deleting DNB event on heifers.
- Deployed new December 2021 USA and Alta sire files.
- Fixed a problem using ABSORB on some cowfiles.

Expand these additional topics to browse new online help content!

New Support FAQ: Find Unused Animal ID Numbers

Did you know you can search the DairyComp database for ID numbers you're not currently using? You can even search across multiple number ranges and print the results in a report. See **Find Unused Animal ID Numbers** to learn more!

DairyComp 305 - Version 21.11.0

Release Date: 12/14/21

This update contains some continuing work on CowCare, Embryo Transfers, and miscellaneous changes and bug fixes. As always, we continue to work on PULSE integration issues as they arise.

Finally, we've spent a lot of time completing the final work on the new non-Java Web Updater which is being tested in this version.

Expand the topics below to learn more about the updates in this release:

CowCare

- Fixed a problem on the Protocols page where protocols would not display after exiting CowCare
- Introduced a drop-down/auto-populate vet in the Documents tab
- Fixed the date display format to DD/MM/YY in some reports and views

PCC

- Included support for Scanner Device ID

PULSE

- Reduced the size of PLCLient backup files to save disk space
- Limited the PLCLIENT to only send to PULSE from a specific cowfile (to avoid potential problems with LOGON use)

Sires

- Fixed a problem with some sires getting changed from conventional 200 to Sexed 777 in SEMINV
- Updated the Italy sire file
- USA Sire file will be released on 12/15

General Enhancements and Changes

- Lely/DC: Fixed SavePregnancy to update the event in Lely, even if out of sequence
- Lactanet: Revised 'TagReplaced' in DairyComp traceability
- Fixed client task tables from being changed in DC Client when the client task table option is enabled
- Removed old and outdated vet reminders from the Vetlist
- Fixed a problem with the online help menu option selecting the tab links to the old reference guide
- Fixed a problem with improperly importing tech tables in ABSORB
- Fixed missing remark/calf info when a cow is sold on fresh date
- Fixed a problem where CONNECT would not automatically import newly downloaded genomics DNG files
- Fixed a problem where having an ETDAM in the archive file could leave the program logged into that archive file

Expand these additional topics to browse new online help content!

New! Get Started Help Content

Information stored in DC305 is comprehensive and easily accessible. In moments, you can determine whether an animal is fresh or dry, when they changed pens last, and if they're due for a check-up with a veterinarian.

Our new Get Started! section is a great resource for new and experienced users alike. See **Getting Started with DairyComp** to learn more!

New! Vet Procedures Help Content

Our new help topic explains all you need to know about vet codes, generating vet lists, and entering vet exam results. See **Veterinary Procedures** to get started!

New Support FAQ: Remove DNB Event

Need to know how to remove a Do Not Breed (DNB) event from a CowCard. See **Remove a Do Not Breed (DNB) Event** to find out!

New Support FAQ: View Energy Corrected Milk Values

Did you know that Energy Corrected Milk (ECM) is a value that can help you evaluate your herd's productivity and efficiency? Our new FAQ explains this helpful calculation and how you can view it in your reports. See **View Energy Corrected Milk (ECM) Values** to learn more!

New Support FAQ: Common Terms and Commands

The DairyComp online help now includes a quick reference page for commonly used terms and commands. The page also includes handy, downloadable PDFs. See **Common Terms and Commands** to check it out.

DairyComp 305 - Version 21.10.0

Release Date: 11/10/21

For this update, the focus was on making continued improvements to CowCare, Embryo Transfer handling, and importing the genomics files. And while we continue to

work on PULSE integration, we've also been working hard to make updating to new releases of DairyComp easier in the future.

Finally, we fixed some bugs and added features requested by customers.

In our next release, we'll have a long-awaited replacement for the aging Java web updater.

CowCare

- Added the ability to import protocols not associated with an event (such as OvSynch)
- Deployed new and updated drug files
- Fixed a bug in the inventory adding process
- Handled a problem maintaining proper premise ID as pens change
- Fixed adding drugs to protocols by going through Alter\7
- Fixed inventory counts for protocols with multiple drugs

Embryo Transfer

We investigated how parentage is assigned to calves based on embryo transfers, whether recorded in DairyComp or in PULSE. Based on that, we fixed a few things and added internal coding of ET breedings as well as better logging for these breedings.

- Fixed setting MGDID and MGSID on calves resulted from embryo
- Fixed wrongly setting DID to zero on calves if the donor is not an active record in the cowfile
- Properly assign donor info to calf if the donor was sold prior to calf being born
- We now log ETBRED breedings as such in the activity log

Pulse

- Fixed a bug where the PULSE syncing would stop due to an error
- We are working on improvements for handling event edits and deletes made on PULSE. This is still in progress awaiting PULSE improvements
- Support importing Fresh Table changes made on PULSE

General Enhancements and Changes

- Fixed problem with some variations of the SIRES\M command to import mating recommendations
- Finished work on the “Type Of Death” item used in Germany
- Support sending DNB to Lely
- We now expose the setting for disabling ID changes in genomic imports, in the Genomics import command. This allows skipping any ID changes (eartags, EIDs, and dam info) from being made by the import
- Allow transferring Protocol details along with Protocols in ALTER\7I import
- We added a setting in SETUP to have a MOVE event created for any PEN change, not just MOVE events. Besides seeing the pen changes on the CowCards, this helps with some of the heifer ranch needs
- Allow Academic versions to access parlor data file
- We added configuration of a “Scan Device ID” item for the PCC refresh, so PCC can send back custom scan IDs (will require future PCC support)
- Fixed bug in Selective Dry Cow Therapy installer
- Fixed erroneous "Duplicate item" message in ALTER\2 caused by deleted items
- Fixed Previous Lactation page display in Russian CowCards
- Store Russian cowfile folder files DCITEMS and DCEVENTS in backups
- We removed the extraneous and repetitive activity log message for failed internet connections

New Reports Help!

The DC305 online help has been updated with new information to help you generate and work with reports, both in the command line and the DC305 menu. See **Reports** and **Command Line** to find out more!

New FAQ: Add a Treatment Protocol

Need to add treatment protocols to DairyComp? See **Add a Treatment Protocol** to learn more!

New FAQ: Toggle Auto Print

Want to toggle the DairyComp Auto Print feature on and off? See **Toggle Auto Print** to learn more!

New FAQ: Assign Tech Team to Events

Did you know you can assign an event to a specific technician (or a group of technicians)? See **Assign Technician Teams to Events** to learn more!

DairyComp 305 - Version 21.9.0

Release Date: 10/13/21

This release contains many changes across the program, with numerous improvements and bug fixes. A lot of work was also done with CowCare.

For PULSE Platform, issues were fixed regarding the FRESH date, as well as some CAR code improvements. Also, we continue to work with the PULSE Platform team to improve the DairyComp to PULSE Platform integration.

Some things with the genomics imports have been fixed and enhanced, including setting (or sometimes *not* setting) DID, MGSID, and other items based on the incoming data, and we made it easier to see these changes.

Also, we are working on improving the Embryo Transfer process, addressing needs for both DairyComp and PULSE Platform.

Finally, as always, we've made many improvements and bug fixes based on requests from customers and the support team.

CowCare Enhancements and Changes

- The multi-premises CowCare problem has been addressed so that CowCare now uses the correct premises when there are multiple premises and premise by event type.
- Fixed the display of the delivery document in the Usage screen. Users now always see the same delivery documents in the same order.
- Changed CowCare so it no longer gets the event/protocol records from the cowfile. The old process was very slow for larger dairies. Now when a user creates an event in DairyComp, a record is created so that CowCare can see the changes without looking into the cowfile. This process, which previously took upwards of 10 minutes, now takes 15 seconds.
- Fixed "Drug Lot" selection on the Inspector Report. The Inspector Report was showing the wrong Drug Lot and Delivery Document in some instances. Also fixed the universal ID on the report.

PULSE Platform Changes

- Fixed a minor issue sending CAR code changes to and from PULSE Platform.
- Fixed a problem that occurred when changing the FRESH date in PULSE Platform.

Genomics Changes

- Several improvements have been made to genomics imports. These include properly setting (or *not* setting) the dam ID, dam reg, maternal grand sire, and sire identifications when the import file says the parentage has changed. Also, logging has been improved and simplified so that now it's easier to see what's been changed by the import, and changes are listed in a separate ImportResults.txt file for later review.
- The optional `\X` switch now allows disabling any parentage changes while still allowing import of the genomic results.

Pocket CowCard Changes

- A problem has been fixed regarding Argentinian and other EIDs starting with zero (0).
- A potential problem has been addressed in which using SETDAY prior to a PCC refresh might have caused the automatic date changes from occurring before the refresh completed.

General Enhancements and Changes

- The US, Canadian, UK, and German sires have been updated with new proofs.
- Germany: Entering a DIED event now prompts for type of death and sends this to the HIT.
- Russia: Commands now properly change when an item name is changed.
- Fixed a problem in some European countries where the breeding technicians could not be changed.
- When an EID (EID, REG1, CREG, etc.) is changed in ENTER, the previous EID is no longer put back into the EID manager.
- Improvements have been made for items shared between DairyComp and PULSE Platform for SmaXtec and SCR.
- Fixed the import of scale weights (TruTest and others) when run on a client.
- Sire item type 190 with an Op2 of 11 now shows the stud name (ALTA, ABS, etc.).
- A minor improvement has been made to the display showing errors or conflicts with items in ALTER.
- The SSF SEND (Agsource, DRMS, Ameliocor, ATA, Canada, etc.) now designates embryo transfer breedings, to allow the DRPCs and CDCB to better track them through the system.
- We improved the `PUTCOW\I` command (the DNC cow transfer) to better detect an internet connection failure. Now it will continue attempting to upload the data until the connection is back up.
- For multi-herd users, we fixed the backup file creation to include the proper setup files from the cowfile folder.
- The `FILEIN\C` command has been improved to properly handle quoted fields in CSV files.
- In the previous release we showed the CAR code description in the CowCard event display for SOLD and DIED. That should have been done for SOLD only, as DIED CAR codes are not always supported. This has now been corrected.
- Fixed a problem in the ALTER-Items command where type 173 did not prompt for all necessary information. Depending on the event, type 173 can show a

protocol description, a culture result, or the disposal reason (CAR).

New FAQ: Find Your DairyComp Account and Serial Numbers

Need to locate your DairyComp account number and serial number?

See **Find your DairyComp Account and Serial Numbers** to learn more!

New Help Topic: CowCards

The CowCard is an animal's record in DairyComp and the cornerstone feature of the program; CowCards tell us a great deal about the animal's birth date, sire identification, breeding status, overall health, and much, much more all at a glance. Need some help navigating the many tools and tabs associated with CowCards? See the new **CowCards** reference page to learn more!

New FAQ: Clear CHECK Status

Need to know how to clear the CHECK status from an animal and remove it from a vet list? Let our new FAQ show you how! See **Clear Check (CHCK) Status from an Animal** to learn more.

DairyComp 305 - Version 21.8.0

Release Date: 08/31/21

This release includes significant improvements to CowCare and the PULSE Platform integration, hooks for future embryo transfer capabilities, and overall program performance enhancements. This release also contains several integrations, dozens of bug fixes, and features based on customer requests.

CowCare Enhancements and Changes

- Significantly improved inventory entering.
- Fixed the handling of Shipments vs. Druglots vs. DeliveryDoc.
- Fixed the process for editing drug Units and Route.
- Fixed and simplified (with a checkbox) the CowCare automatic purchase generation.

Cultures Enhancements

- Item type 173 (Remark detail) has been updated to display the Culture Name (such as "S.agalct") when defined with the CULTURE event as the item Op1.

Embryo Transfer Enhancements

- A field is now set in BRED events for embryo implant breedings. For embryos that come from PULSE or from DairyComp, the breeding includes an ETDAM, which is displayed as "ETBRED." The SSF SEND will recognize this and flag the breeding as an embryo transfer in the send file to the DRPC.

PULSE Platform Changes

- Fixed an identification problem in PULSE Platform that occurred when using the PUTCOW command.
- Fixed a problem sending changes to PULSE Platform that occurred when some ALTER tables were modified.

Pocket CowCard Enhancements

- The refresh cache setting (used to speed up PCC refreshes in large herds) is now accessible in the SERVER settings window.

Integrations Enhancements and Changes

- Made various improvements for the Lactanet XML export process.
- Fixed the long lag time in CowCard browsing for SCR activity herds.
- Fixed a problem with Beco milk import caused by a change in the expected Beco file path.
- Fixed a milk import problem for Lely herds using Russian formats.
- Fixed sending submittal data to Zoetis.

General Enhancements and Changes

- Allow defining of the default folder for the GETCOW\I command.
- The August sire proofs for the US have been released and are included in this update.
- A new genomic trait named GRFI (Genomic Residual Feed Intake) has been added to the DNG file.
- Added various foreign translations.
- The program version number now shows in the window when using the STATUS command.
- Events entered with the same event date but with a different protocol are no longer rejected as a duplicate entry.
- Russian herdcodes are now allowable for iTrans.
- Improved ECON\IR projected heifer counts when sexed or male beef semen is used.
- English description CAR codes (e.g., "Low Production") now display for CowCard SOLD events, instead of just numeric CAR Codes.

New Help Topic: Reproduction Codes

Every animal in DairyComp has a Reproduction Code (RC), but do you know their purpose? Did you know than an animal with an RC value of 4 (RC=4) has been bred but does not qualify as "open" or pregnant? See the new **Reproductive Codes** reference page to learn more!

New FAQ: Set Animal Breed

Want to know how to set the breed of the animals on your dairy? **Check out our new FAQ** to learn about the CBRD item, including how to get DC305 to prompt for the breed of new calves during FRESH events.

DairyComp 305 - Version 21.7.0

Release Date: 07/28/21

This release concentrates on bug fixes and feature requests that had been backing up, as well as some important improvements in CowCare, PULSE Platform integration, and

speeding up DairyComp overall.

Performance Enhancements

Working toward the new Version 9 cowfile involved changes that caused DairyComp's performance to slow down. This version of DairyComp addresses that issue so that now DairyComp performs as quickly as it ever did in the past.

Additionally, we have sped up the device refreshes to Pocket CowCard, resulting in a noticeable improvement. A further improvement is currently being tested which can speed it up even more, especially for large herds refreshing multiple devices. If you want to test this improvement for yourself, please ask the support team to set this up on your DairyComp system.

Data Integrity and PULSE Platform

- Continued improvements have been made in the synchronization to PULSE Platform, fixing some internal IDs so they correctly synch after XID and restores are performed.
- A problem with CAR codes and vet shortcuts for multi-herd setups has been fixed.
- Some missing activity logging of commands entered in PULSE Platform has been fixed.
- Fixed incoming PULSE Platform entries for countries where commas are the decimal separator—for example, allowing both "3.8" and "3,8" as needed.
- Added Allflex items support for AI flag (HT4AI).
- Added an option to clear all technicians from the TECHs table (`ALTER\14M`)
- Work continues with AgSource to integrate MyDC on PULSE and DairyComp.

CowCare Enhancements

- Added the ability to reconcile inventory quantity values in CowCare.
- Fixed a bug in which CowCare was not properly saving settings after they were changed.
- An issue with the Low inventory highlight has been fixed.
- The setting for using auto-generated purchases has been changed.
- Fixed reports to show the proper drug lot and delivery document.
- More translations for foreign users have been added.
- An updated master drug file is being released in this version, which also fixes the option for viewing drug labels.

Other Fixes and Improvements

- A bug introduced in the last release involving remarks in commands (e.g., Automast commands) and for remarks that present a menu selection has been fixed.
- Support has been added for the SmartISO requirement for Argentina.
- AFI MSEND now allows for longer IDs for the Group and Tag fields in cow2_upda.dat.
- A backend change caused some disruptions to the iLoop file transfer process. This issue has been resolved.
- An issue caused vetcodes in the ALTER command to fail if there was a blank constraint. This issue has been fixed.
- The default EID item can once again be used as a milk transponder item.
- Fixed allowing Bred technician numbers over 128.
- In EVENTS command listings, the PEN now shows the pen as of the event date, not the pen the animal is currently in.
- Support has been added for the following new fields in the AgriTech DNI Index files:
  - PTAT: PTA Type
  - UDC: Udder Composite
  - FLC: Feet and Legs Composite
  - REGST: Registration Status
  - RHPCT: RHA Percent
  - LI: Lineage
  - CLASS: Classification Scores
- Added support for beta-casein A2 (BCA2) test results in Jersey's DNI file.
- An issue caused the CONNECT\IU command (i.e., the nightly backup) to add multiple tasks to the Task Scheduler when connections failed. This issue has been fixed.
- When a calf is born as a result of an ET breeding, the ETDAM item on the calf is automatically set to the donor ID. An option called *Set ETDAM to dam on ET*

calves has been added to the ALTER Dairy Info command for customers who want to turn it off.

- Improved automatic adding of items for the Dry Cow Therapy option.
- Removed the ability to accidentally assign a system item (like FDAT or CDAT) to a PCC scan item.

DairyComp 305 - Version 21.6.0

Release Date: 06/28/21

This release primarily addresses several maintenance issues involved in synchronizing DC305 with the PULSE Platform. Significant improvements have been made in keeping data between DC305 and PULSE Platform properly matched. These improvements and fixes are listed below.

DC305 Base Program Changes

- An unexpected expiration message has been removed from PLCLIENT.
- Do Not Breed (DNB) events are now properly written to the activity log when selected in ABORT.
- A problem has been fixed that occurred when clicking the notification icon that appears when a new version is ready to be installed.
- An issue has been fixed that caused ISO IDs from Morocco to display improperly.
- The conflict display of CULTURE result between MOLD and Gram+ Bacillus has been fixed.
- BRED sire selection when using SIR5 (last 5 sires) has been fixed.
- Animal updates in Lely have been fixed.
- A problem with beef withdrawal dates being erroneously set for archived animals has been fixed.

PULSE Platform Changes

- Identification problems caused by using the Change ID (XID) event to change primary IDs has been fixed.
- Problems caused by deleting mistaken SOLD and DIED events have been fixed.
- Problems caused by erasing mistaken FRESH entries have been fixed.
- The movement of cows between cowfiles using GETCOW and PUTCOW commands is now being properly handled.
- The process of applying changes to protocols and other items changed within PULSE Platform has been improved.

CowCare Enhancements

- An issue that caused the Inventory button to disappear has been fixed.
- The delay that occurred when selecting a protocol has been removed.
- An issue with CowCare inventory duplicates has been fixed.
- The speed of the CowCare search process has been significantly improved.
- An issue with subtracting drug lot quantities from the correct drug lot has been fixed.
- Users can now choose both the lot document and the delivery document (instead of just the lot document).
- Text input has been added to drug lots to allow users to update the delivery document at any point, rather than just when a shipment is recorded.
- The method by which users choose drug lots has been improved. Now, a multi-field drop-down is used to help users make a better choice.
- When modifying shipments and/or drug losses, users can define both the drug lot and delivery document (instead of just the drug lot).
- Reports have been changed to use the drug lots delivery document instead of the shipment delivery document. Now, the field from drug lots is the one that gets used and updated. The shipment delivery document is only updated once (upon creation). The following reports were updated:
    - Inspector Report
    - Drug Usage by Delivery Document
    - Purchase/Shipment History

New! CowVal Help Content

The DC305 Reference Guide now includes complete online help content for CowVal, a powerful tool for estimating the value of each adult animal in your herd. Our new comprehensive online help will teach you everything you need to know to get started! See **CowVal (Cow Value)** to learn more.

New! CowCare Help Content

Full online help for CowCare has now been added to the DC305 Reference Guide. CowCare is a dedicated veterinary tool for managing the drugs and treatment protocols for your herd. To check out our easy to use instructions, click **CowCare**!

New FAQ: Manage Disposal Codes

The disposal codes you assign to sold animals can easily be updated as needed. See our **new FAQ** for all you need to know about changing and adding disposal codes!

New FAQ: Fix Client/Server Communication Problems

Having trouble establishing communications between your DC305 client or Pocket CowCard and the DairyComp server? Our new FAQ has some simple steps you can take to try and solve the problem. See **Fix Client/Server Communication Problems** to learn more.

New FAQ: Fix a FRESH Event

Have you mistakenly entered a FRESH event and want to know how to remove it? Let our new FAQ show you the right way to correct this common problem! See **Fix a Fresh Event** to learn more.

New Help Topic: Customize the Menu

We've answered some of your most frequently asked questions about customizing your menu in DairyComp 305! You can add commands to the menu, edit existing commands, or rearrange commands to best fit your dairy's needs. See the new **Menu Customization** reference page to learn more!

New Help Topic: Generate Reports

Find the answers to your **most-asked questions about reports** in one, convenient location: the new **Reports** reference page! Visit the new page to learn how to run a report in the command line, run a report from the menu, or some command line basics!

DairyComp 305 - Version 21.5.0

Release Date: 05/25/21

General Enhancements

- New User Management security lockout: The ability to block the INVOICE command to user levels has been added to the User Management feature. (DC-3057, DCSD-643)
- Improved sorting of event technicians: Users can now control the order in which event technicians are prompted for selection. (DC-3004, DC-1075)
- Increased command length limit: Commands entered into DC305 can now exceed 255 characters. This is helpful when additional characters are needed, such as in ITRANS commands. (DC-2897, DCSD-808)
- New genomic trait added: A new genomic evaluation trait, item name GJNS, has been added to be imported from the DNG file. The description for this trait is "Jersey Neuropathy with Splayed Forelimbs." (DC-2824)

PULSE Platform Enhancements

- Issue involving archives animals: An error occurred in which archived animals were listed as active in the PULSE Platform. This issue has been addressed. (DC-3065)
- Deletion issue: An issue has been corrected that caused problems deleting from PULSE Platform. (DC-3042)
- Data synchronization improvements: Various improvements have been made in synchronizing data between DC305 and PULSE Platform. (DC-3013, DC-2989, DC-2986, DC-2985)
- Fix for damaged PLA/PLB files: Separate users reported damaged PLA/PLB files due to power outages. These issues were investigated and solutions implemented to increase security going forward. (DC-2995, DCSD-902, DC-2990, DCSD-899)
- Data clean-up: A user reported unusual special characters being sent to PULSE Platform. The data required clean-up and this issue has been resolved. (DC-2987, DCSD-891)
- Duplicate animal records: An issue has been addressed involving duplicate animal records on the initial load. (DC-2969, DCSD-880)

CowCare Enhancements

- CowCare timeout setting added: A new setting has been added that allows users to close CowCare after a specified period of inactivity. (DC-3053)
- Inventory adjustment issue: It was reported that entering certain purchases and drug losses erroneously affected inventory for that drug. This issue has been addressed. (DC-3056, DCSD-946)
- Check Past Usages button removed: The **Check Past Usages** button has been removed from the Drug Usage window. (DC-3051)
- Delivery document now editable: Users now have the option to change the delivery document and drug lot from the Drug Usage window. (DC-3025)
- Prescription numbers on drug history reports: A new **Prescription** field has been added to allows users to add prescription numbers to drug history reports. (DC-3024, DC-3018)
- Additions made to Inspector reports: Prescription numbers and delivery document numbers can now be added to Inspector reports. (DC-3023, DC-3022)
- Prescription numbers added for new purchases: Users now have the option of including a prescription number when adding a new purchase. (DC-2919, DCSD-827)
- Inventory start-up issue: An issue has been resolved that caused CowCare inventory to start up slowly. (DC-3002)
- Protocols window change: The **Accept the withdrawal values** checkbox now only appears in the Protocols window when CowCare is enabled. (DC-2922)
- Drug inventory issue addressed: It was reported that a drug loss added to the system did not deduct properly from the drug inventory. This issue has been addressed. (DC-3021, DCSD-926)
- New purchase issue addressed: An issue was reported in which new purchases of a specific drug failed to process correctly. This issue has been addressed. (DC-3019, DCSD-924)
- Protocol editing issue: An issue has been corrected that caused difficulty in editing protocols. (DC-3017, DCSD-919)

- Issue viewing newly added drug lots: A user reported trouble viewing a newly added drug lot. The issue was investigated and addressed to the user's satisfaction. (DC-2838, DCSD-760)

Integrations Enhancements

- Lely API enhancement: Synchronization with the Lely API has been improved. (DC-3060)
- Allflex item mapping enhancement: The item Heat Alert Breeding Window (HTHAI) has been added to the Allflex configuration settings. (DC-3047)
- Hoof Supervisor fixed for Italian herds: A fix has been made to allow Hoof Supervisor data to be imported into Italian herds. (DC-3043)

Pocket CowCard Enhancements

- PCC PIN code validation: DC305 now supports the entry of numeric PIN codes to control who can refresh their devices. This change only applies to PCC Android users and does not affect Windows Mobile users. To see this option, enter `SERVER\3` in the command line to open the Server Configuration window and then select the PCC Pin Codes tab. Look for updates about this feature in the **PCC Release Notes** to learn more. You can also view a video about this feature **here**. (DC-2902, DCSD-422, PCC-880)

Bug Fixes

- Fixed graphing error: An error caused protocols to fail to display in EGRAPH. This error has been fixed. (DC-3067, DCSD-956)
- Corrected technician selection issue: An issue has been fixed that caused problems when trying to choose and move technicians in the Technicians and breeding codes window. (DC-3052, DCSD-945)
- Fixed manual entry of AutoMast Culture results issue: An issue caused the wrong treatment protocol to be assigned to animals when manually entering AutoMast Culture results. This issue has been fixed. (DC-3048, DCSD-944)
- Addressed Bred Hour display problem: An issue has been fixed that caused Bred Hour to fail to display in the activity log. (DC-3039, DCSD-921)
- Fixed GZABT trait loading error: An issue caused the GZABT trait to follow to update when the DND file was loaded from VAS iLoop. This issue has been addressed. (DC-3028, DCSD-930)
- Fixed milk units update issue: An issue has been corrected that caused milk units to fail to update correctly. (DC-3026, DC-2984)
- Corrected item description save problem: An issue caused item descriptions to fail to save in a cowfile version 5. This issue has been corrected. (DC-3016, DCSD-928)
- Addressed issue in LAME graph: An issue has been fixed in which the LAME graph failed to properly show protocols. (DC-3007, DCSD-914)

New Help Topic: Pen Management

We've answered some of your **most frequently asked questions** about working with pens in DairyComp 305! You can add, modify, and delete pens, set up pen capacity, and even move an entire pen of animals to another pen. See the new **Manage Pens** reference page to learn more!

New FAQ: Add Test Day Data

Do you need help loading the test day data for your dairy into DC305? Our new FAQ provides an overview of the process and walks you through the different options available. See **Add Test Day Data** to learn more!

New FAQ: Managing DC305 User Accounts

If you need help adding, editing, or removing your user accounts, or if you're not sure how to define user access levels, check out this new FAQ: **Set Up and Manage Users**

New FAQ: Load EIDs into the EID Manager

Do you have a list of Electronic IDs (EIDs) that you want to add to DairyComp? See the new FAQ to find out how to add your EIDs to your EID manager: **Load Electronic IDs (EID)**

New FAQ: Change Font in Printed Reports

Did you know you can customize the font for your normal and compressed reports? See the new FAQ to find out how to change your font settings, and learn the difference between normal and compressed reports: **Change Font in Printed Reports**

DairyComp 305 - Version 21.4.0

Release Date: 04/08/21

General Enhancements

- SEND\X command update: The SEND\X command now generates a standard set of batch files, regardless of herdcode. These batch files are now correct for uploading to the regional anagraphical system (BDR) for all regions of Italy. (DC-2949, DCSD-847)
- Improved client/server integration between 2x and 3x systems: An improvement has been made to allow DairyComp to more seamlessly interact when sending milk weight data from clients to servers with different time in milk settings (i.e., 2x vs. 3x). (DC-2945, DCSD-838)
- Improved Italian treatment book reports: Reports provided in the Italian treatment book have been modified to improve the accuracy and readability of data. (DC-2920)

Bug Fixes

- Corrected problem of duplicate commands: An issue caused duplicate commands to be uploaded to VAS PULSE Platform. This issue has been resolved. (DC-2974, DCSD-881)
- Fixed GETCOW issue: Addressed an issue that caused an error when using the GETCOW command to retrieve an animal from the archive. (DC-2970, DCSD-851)
- Fixed EVENTS error: A problem has been fixed in which an error message displayed when issuing an EVENTS command. (DC-2968, DCSD-868)
- Addressed FTECH issue: An issue involving milking times with FTECH has been addressed. (DC-2965, DC-2913)
- Corrected lactation data error: An issue caused incorrect lactation data from DairyComp to be uploaded to VAS PULSE Platform. This issue has been corrected. (DC-2958)
- Fixed items anomaly: An issue caused items on CowCards to display intermittently. This issue has been fixed. (DC-2957, DCSD-855)
- Fixed PTA calculation issue: An issue with PTA calculation caused sires to not be in the database. This issue has been resolved. (DC-2956, DCSD-248)
- Repaired twin animal ID issue: An issue has been corrected in which a previously deleted animal ID in DairyComp resulted in a VAS PULSE Platform error. (DC-2935, DCSD-832)
- Fixed CowCare inventory problem: An issue caused CowCare inventory functionality to become disabled. This issue has been fixed. (DC-2876, DCSD-777)
- Fixed CowCare not ready message: An issue has been fixed that caused a "CowCare not ready" message to display when working with DC305 protocols. (DC-2809, DCSD-724)
- Corrected lactation number issue: An issue has been fixed involving the editing of double-digit lactation numbers. (DC-2808, DCSD-724)
- Fixed CowCare inventory totals: An issue has been corrected that caused incorrect inventory totals to display in CowCare. (DC-2575, DCSD-616)

New FAQ: How to Run a Report from the Menu

You may know how to **run reports in the command line**, but did you know that you can also run reports directly from the menu, too? See the new **Run Report from Menu** FAQ to find out how!

New FAQ: Working with Backups

Have you ever wondered how you can manually save a backup, schedule an automatic backup, or even create a backup file in the first place? Then visit the new **Working with Backups** topic to find the answers!

New FAQ: Edit Sire Info

Did you know you can edit your Sire information in your CowCards? See the new FAQ to find out how to enable editing: **Edit Sire Info**!

DairyComp 305 - Version 21.3.0

Release Date: 03/15/21

General Enhancements

- Improved DNL data loading between lactations for Lactanet herds: Loading a new DNL file for a Lactanet herd now clears the data from the previous lactation, removing the need to do so manually. (DC-2708, DCSD-483)
- Updated PODAT logic: The logic involved when uploading the Preg/Open date (PODAT) when a cow is aborted in bullpen has been updated. (DC-2709, DCSD-396)
- Improved communication with Lely integration: The synchronization of data when saving an animal has been improved when using the Lely integration, eliminating the need to first "transfer out" the animal. (DC-2678)

CowCare Enhancements

- CowCare failed to show all premises: It was reported that CowCare failed to show all available premises for a dairy. The issue was traced to a Windows setting and has been resolved. (DC-2857, DCSD-764)
- Lot numbers and sorting: On the Drug Usage page, sorting for the Event column has been improved, and the Lot ID can now be edited. (DC-2801, DCSD-596)
- CowCare drug manufacturer field added: When adding a drug to CowCare, users can now select the drug's manufacturer from a new drop-down menu. This improvement will speed up and improve data entry. (DC-2634, DCSD-617)

PULSE Platform Enhancements

- Missing records in VAS PULSE Platform: An error occurred when attempting to restore Sold/Died animals after reloading herdfiles on VAS PULSE Platform, leading to missing animal records. The system has been updated to address this issue. (DC-2862, DCSD-771)
- VAS PULSE Platform worklists failed to generate: DairyComp failed to properly generate worklists containing animals with Platform IDs that were incorrectly loaded from VAS PULSE Platform. (DC-2861, DCSD-774)

Bug Fixes

- Corrected problem viewing TRACE records: An issue has been resolved that caused the Review Traceabilty tab of the TRACE module to not show current records to be reported. (DC-2870, DCSD-751)
- Corrected wrong registration value suggestion: A problem has been fixed in which DairyComp suggested the wrong registration value when entering new calf data. (DC-2867, DCSD-779)
- Fixed printing errors related to paper sizes: Addressed an issue in which the last version of the dcwin.exe file caused intermittent paper size issues when printing. (DC-2827, DCSD-756)
- Corrected error displaying different breeding technician names: Resolved an inconsistency in the names of breeding technicians displayed by the system between the latest version and previous versions. (DC-2770, DCSD-723)
- Repaired incorrect TRACE module event dates: An issue has been fixed that caused animals entered into DairyComp using the `ADD` command to be recorded in the TRACE module with the wrong event date. (DC-2778, DCSD-732)
- Fixed incorrect milk weights for 2x cows in 3x herd: Incorrect milk weights were reported for cows being milked 2 times daily in a herd where other cows were milked 3 times daily. This calculation error has been fixed. (DC-2783, DCSD-658)
- Fixed incorrectly reported milk weights: A calculation error caused milk weights to be reported incorrectly for herds in Germany. This calculation error has been fixed. (DC-2784, DCSD-548)
- Repaired CLEANUP command data errors: Running the CLEANUP command on DairyComp v7 cowfiles caused errors in some ALTER data elements. The problem was traced to a particular cowfile sorting method and has now been repaired. (DC-2894)

New FAQ: How to Add Commands to the Menu

Did you know your DC305 menu is highly customizable? You can add your most commonly run commands and reports to it so that anyone in your dairy can run them quickly and easily with a couple of mouse clicks! See the new **Add Commands to the Menu** FAQ for instructions!

New FAQ: How to Add New Events to DC305

DairyComp 305 allows you to create and save custom events that are specific to your dairy. See the **Define Custom Events in DC305** FAQ to learn how!

New FAQ: How to Set up Pen Capacity

This is a common task that is easy to perform! See the new FAQ for instructions.

New FAQ: How to Refresh a DairyComp Client

Has your DairyComp client stopped communicating with your DairyComp 305 server? VAS Support is often asked for help with this issue! Check out the solution in the new **Refresh a Client** FAQ!

New FAQ: Confirm that DairyComp is Backing Up to the Cloud

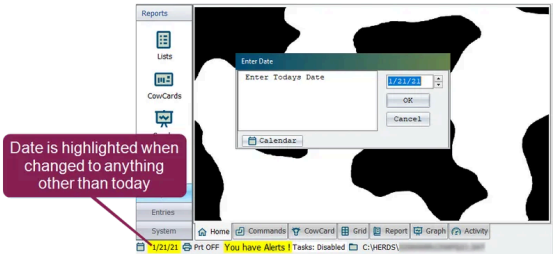
Do you know whether your DC305 data is scheduled for regular back-up? The VAS Support team has some quick steps you can take to be sure! See this new FAQ for more information about this solution, as well as other answers for your most-asked questions about backups: **Backups FAQ**!

DairyComp 305 - Version 21.2.0

Release Date: 01/27/21

General Enhancements

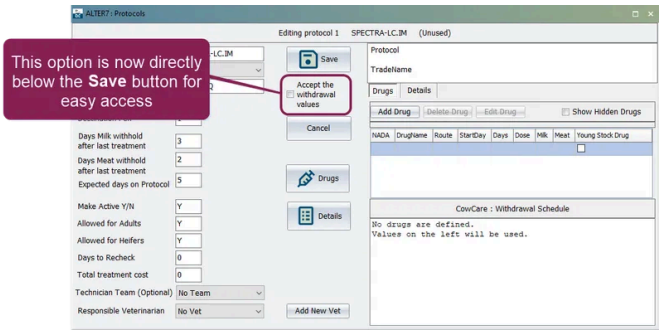
- Improved sire name display in BREDSUM report: We expanded the width of the sire names column in the BREDSUM report so that it does not truncate the names. (DC-2735, DCSD-701)
- Highlight on current DC305 date: When you set the DC305 current date to any day other than today, that date is now highlighted in yellow at the bottom left of the screen as a visual warning that it is not the current date. (DC-2735, DCSD-701)



- Expanded the number of lines in an INVOICE: To accommodate larger herds and multiple sites, we updated DC305 invoices to include more lines in one invoice. (DC-2665, DCSD-671)
- Adjusted SUM column width: We have adjusted the 1st column of the SUM report to fit the full width of the contents (rather than truncating it). (DC-2642, DCSD-356)
- Added CLEANUP command switch for died/sold animals: You can now type `CLEANUP\E` in the command line to clean your archived died/sold animal records. This is a shortcut to running the CLEANUP command and then clicking the **Fix missing Sold/Died Cows** button in the Cleanup dialog box. (DC-2604, DCSD-356)
- GUIDE now available in French: A French translation for the GUIDE is now available. (DC-2604, DCSD-294)

CowCare Enhancement

Usability improvement for editing protocols: When you are editing a CowCare protocol, you can now find the **Accept the withdrawal values** checkbox at the top of the page, directly below the **Save** button.



Lely Integration Enhancements

- Revised processing for transferred animals in T4C: We have updated the system to ensure that animals that are set as "transferred" but are still on the farm are able to be saved via the Save-Animal command. (DC-2604, ISD-180)
- Communication with Taurus prevented for Lely API users: If your dairy has converted to using the Lely API, DC305 will no longer send data to Lely through Taurus. If you want to revert back to sending through Taurus, you can change the setting in `SETMILK\INSTALL`. (DC-2172)

Enhancements for Bull Records

- Updates for adding bulls to events:
  - If a user attempts to add a reproductive event to a bull when it only applies to a female animal (PREG, FRESH, HEAT, etc.), the system now displays a warning prompt that prevents the add. Bulls can still be added to other events that do not impact reproductive status. (DC-2144, DCSD-321)
  - We have updated the system to preserve the RC of an animal when it is added to a DNB event. This ensures that bulls are not mistakenly converted to a heifer in the system if the animal is sold and restored. (DC-2142, DCSD-313)
- Updates for restoring bull records: When restoring the record for a bull that was previously set as SOLD, the system now checks for whether the animal is RC=8 (indicating that it's a bull), and if so, it preserves that setting rather than restoring the animal to RC=0 (indicating that it is a female restored to the herd). (DC-2033, DCSD-269)
- Updates for restoring freemartin records: When restoring the record for a freemartin that was previously set as SOLD, the system now retains the animal's RC=1 setting (NO BRED) instead of converting her to RC=0, which indicates that she can be bred. (DC-1717, DC-1884)

New FAQ: How to Delete a Test Date

VAS Support is often asked how delete a specific test date from an animal's record. Check out the new **Delete a Test Date** FAQ for instructions!

Bug Fixes

- Corrected COWVAL heifer count: Repaired an issue that caused bulls to be included in the COWVAL heifer count. (DC-2742, DCSD-682)
- Repaired CHKFILE report format: Repaired an issue that caused data to display in the wrong columns when adding attributes to the CHKFILE report. (DC-2706, DCSD-699)
- Corrected EGRAPH display when double-clicking to view details: When you double-click on the values in an EGRAPH report, the system now displays the correct detailed data. (DC-2539, DCSD-584)
- Repaired report time format: The system now displays the report time in 12-hour AM/PM format. This fix applies only if you use the SETUP option to show the time on your reports. (DC-2573)
- Repaired date format in the Alter 6 grid: Updated the Alter 6 grid to display dates in the standard DairyComp format, with no leading zeros. For example, January 3, 2021 is now displayed as **1**/**3**/21 rather than **01**/**03**/21. (DC-2688, DCSD-680)

DairyComp 305 - Version 20.11.0

Release Date: 01/04/21

Genomics Enhancements

- Updated and calculations for DNG file import: DC305 has been updated to support the current JPI and JUI formulas.

  > **IMPORTANT:** You must reload your genomics to apply the correct values from these calculations.

- Support for , , and genomic trait imports: We updated DC305 to import the new HLV (Heifer Livability), FS (Feed Saved), and MSP (Milking Speed) genomic evaluation traits that the is adding to the DNG file in December 2020.

Bug Fixes

- Repaired CowCare inventory duplication: The system no longer creates a new inventory item when you create a new protocol. (DC-2630, DCSD-655)
- Repaired "no internet" message: The system no longer displays an incorrect error message saying that the user is offline. (DC-2616, DCSD-652)
- Restored check for milkweights with unsuccessful milkings (Lely Integration): DC305 now properly checks to see if there are milkweights in unsuccessful milkings so that they are not skipped. (DC-2612, ISD-172)
- Fixed lactation card printing issue: When printing lactation cards with Exam\7, the system no longer prints the previous cow's lactation graph on the 2nd cow's card. (DC-2594, DCSD-613)
- Updated the EGRAPH legend: Improve the ability to The legend that describes the colors and items in an EGRAPH now shows proper names when graphing item type 65. (DC-2538, DCSD-599)
- Repaired display of tech name in events: If a technician's name is defined in the DC305 technician table, the tech's name is now properly displayed in events instead of the tech number. (DC-2505, DCSD-590)
- Repaired BULLPEN date for bulls: When a bull is placed into a bullpen, that animal no longer gets an HDAT assigned to it. (DC-2143, DCSD-306)
- Removed unnecessary SEND\X prompts (for Italian herds only): We repaired the SEND\X command to prevent additional prompts. (DC-2640, DCSD-666)
- Updated system logic for sending twin data to the Platform: DC305 now sends all data for twin animals to VAS PULSE Platform, rather than dismissing additional twin animals as duplicates. (DC-2578, DCSD-628)

New FAQ: How Do I Restore the Record for a Sold or Died Animal?

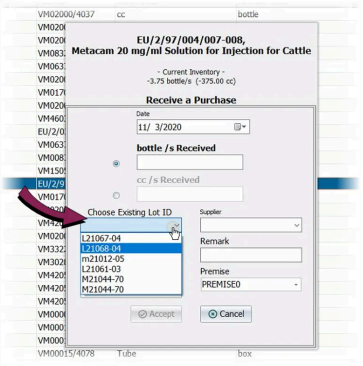
Learn how to restore a record for a SOLD or DIED animal: VAS Support is often asked how to restore the record for an animal that is still on your dairy but has been marked as SOLD or DIED. Check out the new **Restore an Animal Record** FAQ for instructions!

DairyComp 305 - Version 20.10.0

Release Date: 11/11/20

CowCare Enhancements

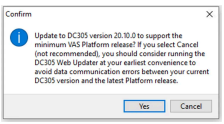
- The CowCare inventory module was updated in the following areas:
  - CowCare inventory totals now round to the nearest tenth. (DC-2530, DCSD-578)
  - Repaired an issue that caused the system to hang when changing inventory values. If you receive an error message when changing inventory values, the system now allows you to close out of the message and accept your updates. (DC-2554 , DCSD-605, DCSD-627)
- Repaired processing error when approving pending changes: When you open CowCare Protocols, approve pending changes, and then exit CowCare, the system no longer gets stuck in a processing loop. It processes your changes and closes properly. (DC-2523, DCSD-594)
- Repaired the Lot ID field: The drop-down menu in the Lot ID is now displayed properly again (DC-2580, DC-2583, DCSD-631).



- Ability to designate an alternate item type for the animal's permanent ID number: To accommodate herds that do not always use item type 5 (EID) to define an animal's ID, we added the ability to specify the animal's permanent ID value as either **EID**, **REG**, or **REG1**. (DC-2581)
- Repaired server-to-client sync on the Protocol table: Repaired an issue that prevented DC305 client machines from receiving server updates on the CowCare Protocol table. (DC-2575, DCSD-616)
- Restored the ability to add drugs to an ad-hoc protocol: When you attempt to add a drug to a user-defined protocol in the Protocols table, the system no longer triggers an error message that prevents you from proceeding. (DC-2555, DCSD-608)

Other Enhancements

- New prompt when a DC305 update is required to support updates in the VAS PULSE Platform: We have improved the message that DC305 displays when an update is required to support the latest VAS PULSE Platform (see below). We have also updated the system to automatically proceed with the update when you select **Yes**.



- Client updates remain in sync with server updates: We updated the DC305 client updater (DCLaunch.exe) to ensure that DC305 client versions are updated when the DC305 server is updated to a new version. (DC-2482, DCSD-563)
- Increased the number of milking schedules in FTECH: FTECH now allows 12 milking schedules instead of 6. (DC-1668)
- Expanded the Lactanet RECEIVE command: The RECEIVE command for Lactanet cowfiles (cowfile type =14) now imports the following additional data (DC-2378, DCSD-483):
    - MEFAT: Current 305 ME Fat
    - MEPRO: Current 305 ME protein
    - STDM: Last test day Standard milk (kg\d)
    - DPD: Dollars per day of life
    - M365: Current 365 day Milk (kg)
    - F365: Current 365 day fat (kg)
    - P365: Current 365 day protein (kg)
    - MVAL: Current lactation milk value

Bug Fixes

- Repaired the ability to edit incorrect Premises ID: Resolved an issue that prevented users from editing incorrect Premises ID for Canadian herds. (DC-2582, DCSD-630)
- Repaired Afi MSEND: The system has been updated to support 3-digit pen numbers when executing an MSEND \H command for the AfiMilk integration. The system response no longer deletes the last digit from the ABORT date. (DC-2532)
- Repaired SUM command for item type 145: When you run a SUM BY command for item type 145, the system now correctly displays the associated technician. (DC-2494, DCSD-574)
- Added support for alternate decimal symbols in the Smartbow/DC305 integration: Updated the system to support regional data values that use alternate decimal formats (for instance, a comma "," instead of a decimal "." symbol). (DC-2525, ISD-160)

New DC305 Online Help!

Welcome! Browse the sections below to learn more about DairyComp 305, as well as what's new in DC305, Pocket CowCard, and FeedWatch (on-premise)!

See the topics below to learn more!

- What's New in DC305 and PCC (this topic): Check here for the latest DC305 and PCC Release Notes.
- **DC305 Support FAQs:** Check out the most Frequently Asked Questions (FAQs) from the VAS Support team!
- **New DC305 Reference Guide:** We'll use this page to post reference topics for commonly used DC305 features. Check out the first one here: **Command Line**!

DairyComp 305 - Version 20.9.0

Release Date: 10/08/20

Enhancements

- Ability to pass decimal dosages to CowCare: When you enter a drug dosage as a decimal in the remark (REM) of an event, CowCare can now receive that value from the DC305 event remark. Previously, CowCare could only receive drug dosages as whole numbers. (DC-2464, DCSD-558)
- Ability to calculate BREDSUM and EGRAPH before the EDAT: By default, BREDSUM and EGRAPH only calculate events that were recorded on an animal's record after its Enrollment Date (EDAT). You can now use the new \K switch for EGRAPH and the \H switch for BREDSUM to include data from before the EDAT. (DC-2407, DCSD-476)
- SDCT tab added to Guide: For users who have SDCT (Selective Dry Cow Therapy) enabled, you will now see a SDCT tab in the Guide. (DC-2267)
- DairyTrace reporting for Canadian dairies: DC305 now allows reporting to the DairyTrace database. (DC-2245, DCSD-387)
- DC305 event edits now synchronize with the Platform: We improved system logic to allow events changed in on-premise DC305 to be updated in the Platform. (DC-1930, DC 1931, DC 1932, DCSD-233, DCSD-234, DCSD-240)
- Lely/DC305 Integration updated to prevent SETMILK date change: For Lely API integrations with DC305, a new \S switch is now available in the MILK command to address the situation where an additional parlor's milk integration imports milk weights. The \S switch prevents milk weights from being imported with today's date until the final shift of the milk parlor for yesterday is completed. (DC-2391, ISD-141)

Bug Fixes

- Repaired ECON\H item mapping: When you run ECON\H and proceed with option 4 (Import Easy Registration File), the item mapping selections (REG, SREG, DREG, etc.) are now visible again. (DC-2501, DCSD-591)
- Repaired ability to edit a protocol description: Restored the ability to edit a protocol description for that protocols that are in use. (DC-2481, DCSD-562)
- Repaired Monitor\C printing: Repaired a bug that prevented the MONITOR report from printing correctly in compressed format. (DC-2472, DCSD-535)
- Repaired missing DC305 IDs in Platform: Improved the DC305-to-Platform initial load to ensure that all DC305 IDs are properly uploaded to the Platform. (DC-2462, DCSD-560)
- Repaired UK sire names: Repaired an issue that caused some incorrect sire names in the UK sires file. (DC-2452, DCSD-550)
- Repaired a TRACE data bug: DC305 now correctly receives TRACE data for animals that are entered through a FRESH event on the client. (DC-1618)
- Repaired an issue for the Boumatic Integration specific to dairies with 4 milking cycles: DC305 now properly loads milk for the 4th milking from Boumatic Robots. (DC-2496, ISD-153)
- Repaired sendpar to afi: Repaired an issue that caused sendpar to cause an error in the afi when animals that were XIDed had the same previous ID. (DC-2490, ISD-577)
- Repaired DC/ICE cow alert (Send to Cow alert was unsuccessful): Repaired an issue that caused all sending in the ICE interface to fail. (DC-2477, ISD-151)

DC305 - Version 20.8.0

Release Date: 09/08/20

New Feature! CowVal 1.5

This DC305 update delivers some exciting enhancements to the CowVal feature!

For herds that are cross-bred or that contain multiple breeds, we are aware that it was difficult to base a CowVal score on quantity alone. With the CowVal 1.5 update, you'll continue to see details on the quantity of the milk produced, but now when you run COWVAL\F, you'll also receive information on **Milk Fat** and **Milk Protein**.

Adjusting the current CowVal formula to include Protein and Fat benefits farms that would like to use COWVAL to set a score on their cows, as these components are key values in the milk check.

Read below to learn **what changed** and **how to set up CowVal 1.5**.

What Changed?

CowVal 1.5 takes into account the animal's contribution of **Milk Fat** and **Milk Protein**. As you can see in the example to the right, we have added the **Protein Price** and **Fat Price** to give a value of the components that the animal is contributing. We also have added fields for **Lactation Fat** and **Lactation Protein** values to serve as defaults if a cow doesn't have test day data. The **Mature Equivalent** values are based off of data from your herd; with the use of this formula, we can better compare animals in a multi-breed herd.

How to Set Up CowVal 1.5

To use the new CowVal 1.5 feature, enter the command COWVAL\F and then select **Edit Parameters** to access the updated table. The items used are still CWVAL and PGVAL and will be populated with the new formula. To schedule the recalculation of these items, you can use the command **COWVAL\FG**.

Enhancement: Ability to Change CowCare Drug Names

Added the ability to change a drug's name in the CowCare Inventory. (DC-2350, DCSD-469)

Bug Fixes

- GETCOW\I Protocols importing incorrectly: Repaired an issue to ensure that protocol numbers import correctly with GETCOW\I. (DC-2380, DCSD-375)
- Chinese - Item list from Alter\2: Fixed item name issues when an item is edited in Chinese. (DC 2352, DCSD-475)
- DCTEACH - Protocols do not allow editing and not all features are there: Enabled CowCare on DCTeach and repaired an issue to ensure that protocols can be edited. (DC-2351, DCSD-479)
- Inventories not matching between PULSE Platform and On-Premise DC305: Fixed several issues that were allowing for additional cow records in the PULSE Platform because previous lactations were being flagged as a separate cow. (DC-1934, DC-2357, DCSD-231)
- Platform FRESH entry posting duplicates and XIDs to DC305: Repaired an issue that caused XIDs and duplicate calf IDs when entering a FRESH event on the Platform. (DC-2444, DCSD-514)
- CowCare modify contacts error: Repaired a bug that caused an incorrect warning message and sometimes prevented the use of CowCare.
- Printer and Page header issues: Several printing issues were resolved. (DC-2346, DC-2369, DC-2364, DCSD-488, DCSD-489, DCSD-491)

For release notes prior to August 2020, please see the following pages on the VAS website:

- **DairyComp 305 Release Notes Archive**
- **Pocket CowCard Release Notes Archive**

9. vas duplicate

10. Testing diacritics

> - început
> - cîntec

- muștar
- șiret

11. CrossRiver test page 3

Relevant webhooks

For this endpoint be sure yoxu have **Accounts, cards and payment events** for this webhook:

Add file

File URL

Add File

- **Rtp.Payment.Received**
- **Rtp.Payment.Expired**
- **Rtp.Payment.Rejected**

API reference

Initiates a request for payment (RfP)
Allows the recipient to authorize and trigger an instant payment credit transfer. RfPs can be initiated from:

- Master account
- Deposit account (DDA)
- Associated subledger

Endpoint: /rtp/v1/payments/payment-request

POST https://sandbox.crbcos.com/rtp/v1/payments/payment-request

Request | Response

BODY PARAMETERS

accountNumber `String` `required`

Account from which the RfP is originated

amount `Integer` `required`

Requested payment amount in positive integeral cents. For example, represent $1.00 as 100.

purpose `String` `optional`

Internal CR field used to include a short description or purpose for the payment request. The RDFI does not see this field. 50 characters maximum.

requestedExecutionDate `String` `required`

Date when the payment request needs to be processed by the Debtor. Formatted as yyyy-mm-ddThh:mm:ss[.mmm].

expiryDate `String` `required`

Expiration date and time of the RfP. If no time is specified, it expires at 12:00 AM on the given date. If a time is provided, it expires within one hour after the specified time on the same date. Formatted as yyyy-mm-ddThh:mm:ss[.mmm].

clientIdentifier `String` `optional`

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes. 50 characters maximum.

remittanceData `String` `optional`

A description of the payment request, meaning, the remittance advice. Use this field for any remittance information that doesn't have its own field. 140 characters maximum.

remittanceLocation ▶ `Object` `optional`

Remittance location information

debtor ▶ `Object` `required`

Debtor Personal/Business information. Debtor is the individual receiving the Request for Payment. Please note: For the debtor, name and one additional element like [birthCity, birthCountry, birthdate], [orgLegalEntityId/LEI], OR [addressStreetName, addressCity, addressState, addressPostalCode, and addressCountryare] required as per RfP business rules.

ultimateDebtor ▶ `Object` `optional`

Ultimate Debtor Business/Individual Information. The final party receiving the RfP and responsible for initiating the credit transfer upon authorization.

creditor ▶ `Object` `optional`

Creditor Business/Individual Information. The recipient of the payment, typically the party at CRB initiating the RfP. Depending on the configuration, this information may be populated automatically.

ultimateCreditor ▶ `Object` `optional`

Ultimate Creditor Business/Individual Information. The final recipient of the payment. Depending on the configuration, this information may be populated automatically.

useCase `String` `required`

Use one of the following values for the useCase field:

-BusinessToBusiness
-ConsumerBillPay
-AccountToAccount
-DownPaymentOrFinalPayment
-ConsumerToGovernment

industryCategory `String` `optional`

Use one of the following values for the industryCategory field (optional when useCase is BusinessToBusiness or ConsumerToGovernment):

-HomeUtilities
-TelecomUtilities
-Mortgage
-Rent
-InsurancePremiums
-AutoLoans
-PersonalLoans
-CreditCardPayment
-MediaSubscriptions
-Memberships
-FinancialInstitution
-BrokerOrDealer
-Medical
-OnlineGaming
-RealEstate
-Government

senderID `String` `required`

Use the senderId field (optional when useCase is BusinessToBusiness or ConsumerToGovernment), provided by Cross River and generated by The Clearing House (TCH).

Valid senderID's for Sandbox Testing:
-100000000001
-100000000002
-100000000003
-100000000004
-100000000005

amountModAllowed `Boolean` `optional`

Indicates whether the recipient is allowed to modify the requested amount in response to an RfP. If the field is not provided, the system defaults to true.
If true, the recipient may respond with a different amount
If false, the recipient must match the original amount; any mismatch will cause the payment to be rejected.

```
curl --location 'https://sandbox.crbcos.com/rtp/v1/payments/payment-request' \
--header 'Content-Type: application/json' \
--data '{
    "accountNumber": "2564791479",
    "amount": 20000,
    "discount": 1000,
    "debtor": {
        "routingNumber": "071212128",
        "accountNumber": "2758463448",
        "name": "Terry Tester",
        "birthCity": "Pine Brook",
        "birthCountry": "US",
        "birthDate": "1996-08-09",
        "addressStreetName": "Linwood Ave",
        "addressBuildingNumber": "2115",
        "addressCity": "Pine Brook",
        "addressState": "NJ",
        "addressPostalCode": "07024",
        "addressCountry": "US"
    },
    "creditor": {
        "accountNumber": "2564791479",
        "name": "the creditor"
    },
    "purpose": "Test Payment 2",
    "clientIdentifier": "123456",
    "requestedExecutionDate": "2025-08-10",
    "expiryDate": "2025-08-16",
    "senderId": "100000001",
    "useCase": "BusinessToBusiness",
    "industryCategory": "OnlineGaming"
}'
```

```
{
    "id": "8ee74810-45fe-47ee-9569-b30500b8d6ae",
    "originalPaymentId": "8ee74810-45fe-47ee-9569-b30500b8d6ae",
    "referenceId": "R174OI808VYXJ",
    "accountNumber": "2955057589",
    "clientIdentifier": "123456",
    "amount": 19000,
    "direction": "Outbound",
    "status": "Created",
    "paymentType": "RequestForPayment",
    "source": "Api",
    "transactionAccountContext": "NotSubmitted",
    "debtor": {
        "routingNumber": "011000138",
        "accountNumber": "2758463448",
        "name": "Terry Tester",
        "birthCity": "Pine Brook",
        "birthCountry": "US",
        "birthDate": "1996-08-09",
        "addressStreetName": "Linwood Ave",
        "addressBuildingNumber": "2115",
        "addressCity": "Pine Brook",
        "addressState": "NJ",
        "addressPostalCode": "07024",
        "addressCountry": "US"
    },
    "creditor": {
        "routingNumber": "021214891",
        "accountNumber": "2955057589",
        "name": "Dim Mak",
        "addressStreetName": "400 Kelby St",
        "addressCity": "Fort Lee",
        "addressState": "NJ",
        "addressPostalCode": "07024",
        "addressCountry": "US"
    },
    "network": {
        "clearingSystemCode": "TCH",
        "currency": "USD",
        "endToEndId": "6909c29416f0453fbdfdb30500b8d6ae",
        "instructionId": "20250623021214273T1B19TS07136537534",
        "interbankSettlementAmount": 19000,
        "interbankSettlementDate": "2025-06-23",
```

```
    "messageId": "M20250623021214273T1BDQK15672530466",
    "numberOfTransactions": 1,
    "settlementMethod": "CLRG",
    "businessMessageId": "B20250623021214273T1BPRK46725920502",
    "toParticipantId": "990000001S1",
    "fromParticipantId": "021214273T1",
    "instructingRoutingNumber": "021214891",
    "instructedRoutingNumber": "011000138",
    "headerCreationDate": "2025-06-23T07:12:58.6058948-04:00",
    "messageCreationDateTime": "2025-06-23T07:12:58.6058948-04:00",
    "createdAt": "2025-06-23T07:12:58.6058948-04:00"
```

12. Test folder

12.1. testing missing TOC header

Release notes provide updates on new features and changes that occur on the Cadent Platform as part of a given release. We make new release notes available as soon as a new release occurs.

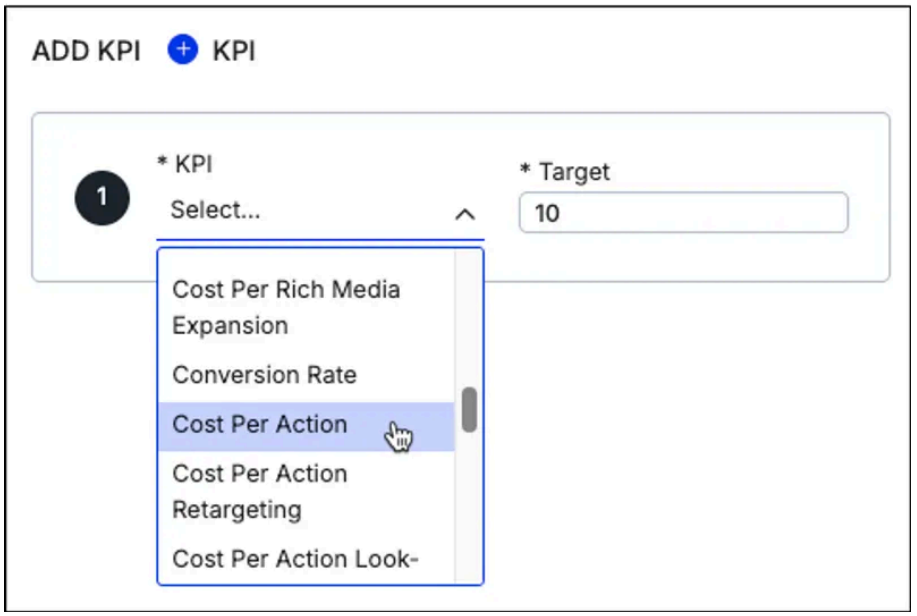
April 21, 2026

Ad Manager Feature

Cost Per Action KPI for line items

Ad Manager now supports cost per action (CPA) models, which are a campaign-specific models created automatically when you select **CPA** as the KPI for a line item, and then map the activities on the campaign.

CPA models are a key tool for driving strong KPI results by providing performance-based models trained on conversion data, like a sale or an app installation. This is different from other models, such as click models, where they optimize towards clicks regardless of whether they lead to a conversion.



Cost Per Action is a line item KPI

April 20, 2026

Ad Manager Features

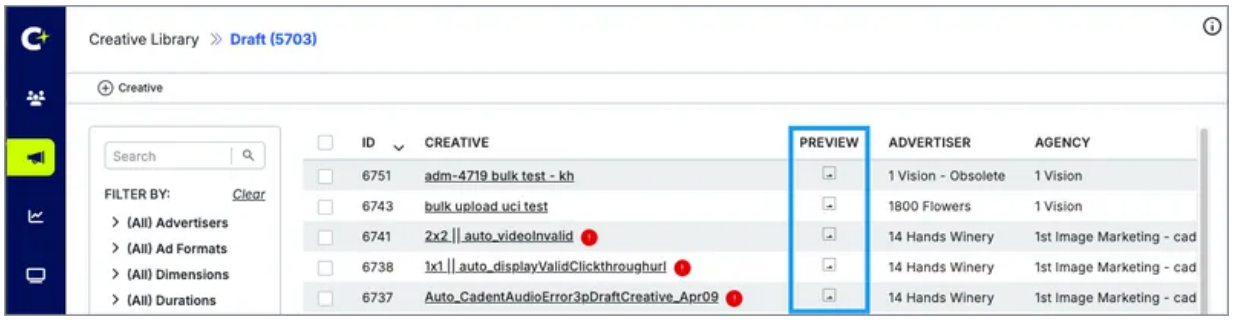
Preview and test your draft creatives

You can now preview and test your draft creatives. This quality control step ensures your draft creatives render and function exactly as intended before you **approve your draft creatives**.

Learn more about **previewing and testing creatives**.

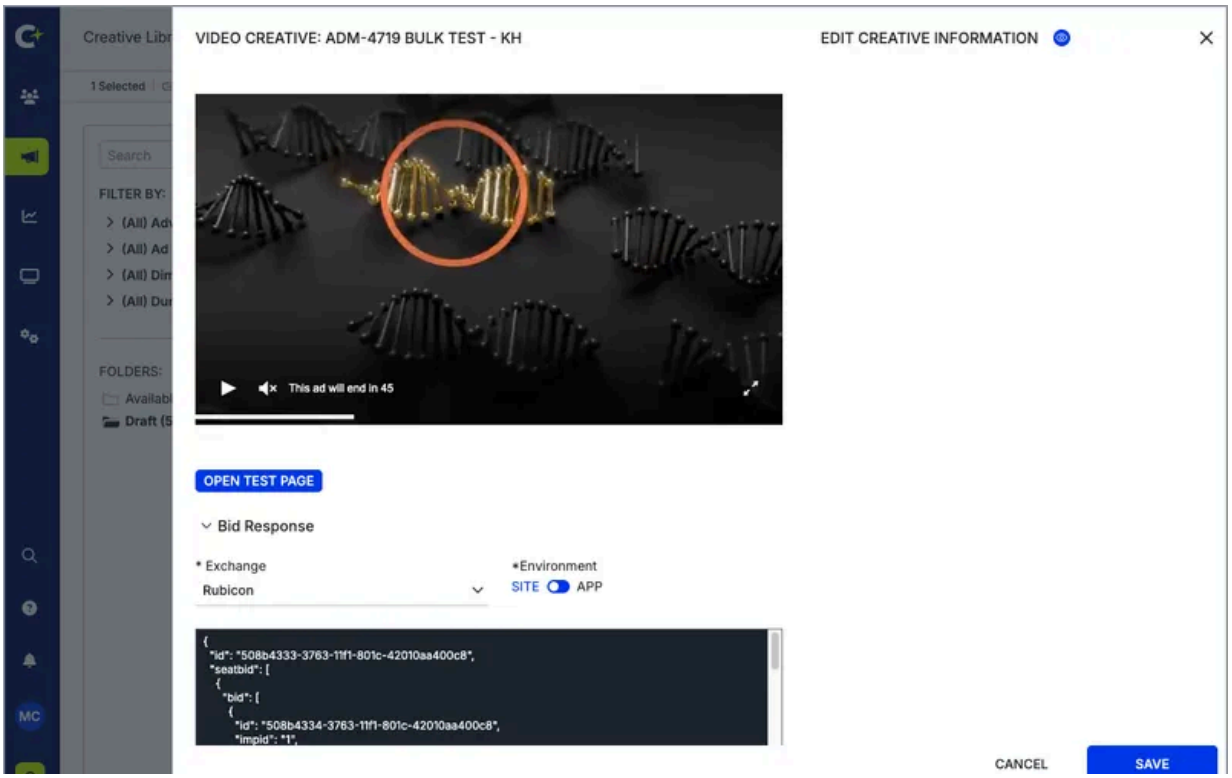
How to preview and test a creative

1. Navigate to the Creative Library, then click the **Preview** icon next to the creative. This opens the preview and test panel.



Preview icon in the Creative Library

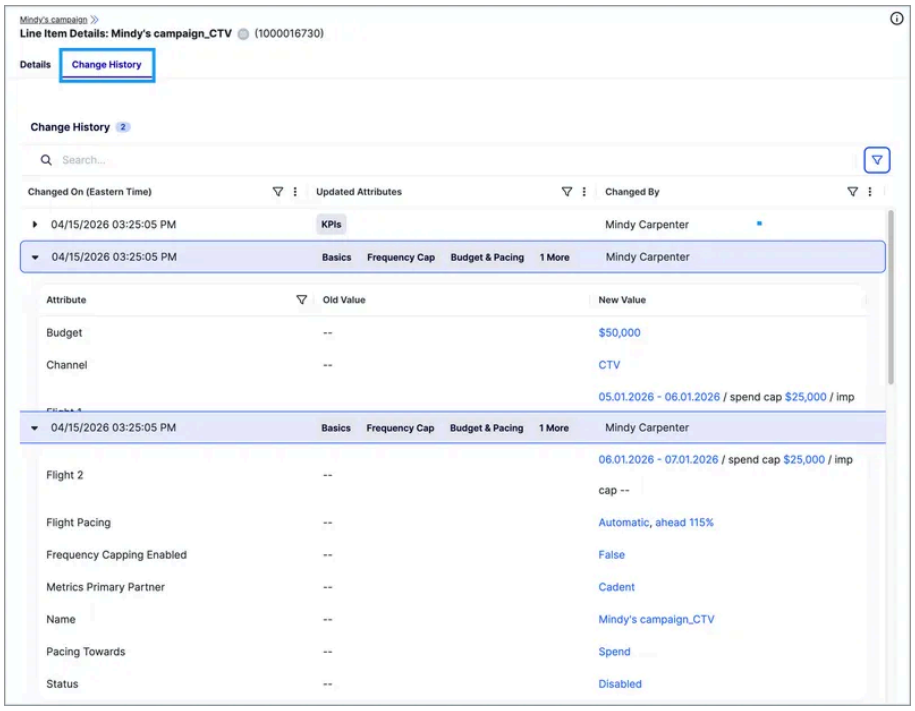
2. Here, you can do the following:
    - Click the creative preview to test the click-through URL.
    - View the bid response.



Introducing line item Change History log

You can now review a complete log of all changes made to your line items. Simply navigate to the new **Change History** tab on the Line Item Details page. There, you will find a detailed log of every modification, including:

- Date the change was made (Eastern Time)
- Attribute that was changed
- Old Value
- New Value
- Who made the change



Change History tab

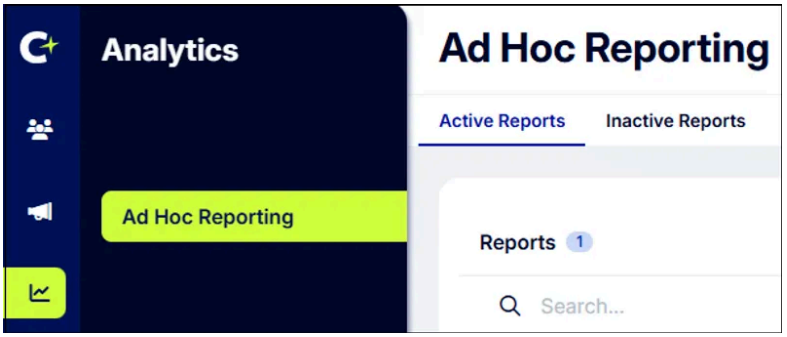
This feature gives you greater transparency and makes it easier to troubleshoot by providing a clear audit trail of all adjustments.

Learn more about the **Change history log**.

Analytics - Ad Hoc Reporting Features

Ad Hoc Reporting has moved

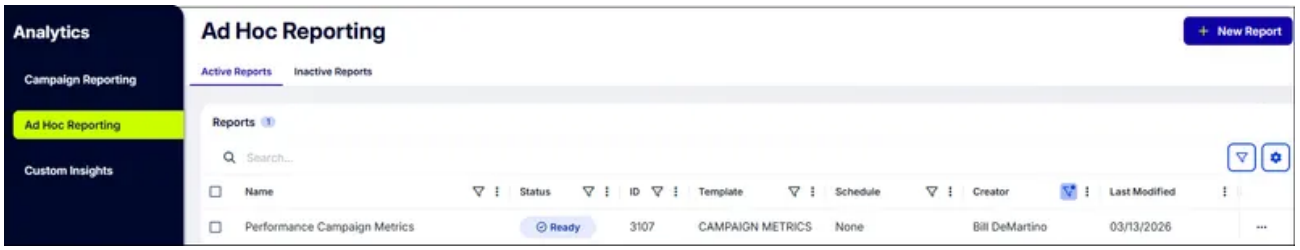
You can now find Ad Hoc Reporting by clicking the Analytics icon on the left-side nav bar. In the past, it was under Ad Manager.



Ad Hoc Reporting page improvements

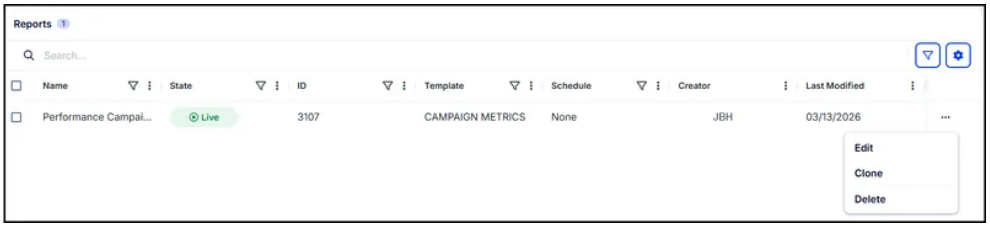
The Ad Hoc Reporting page has been updated to improve it's workflow and efficiency. You have the ability to add or remove columns. More powerful filters provide the

capability to view a specific set of reports and sort them to suit your needs. **Learn more.**



Clone reports for faster changes

On the Ad Hoc Reporting page, you can now clone a report. Cloning allows you to save time when creating a new report by using an existing report as the foundation for a new one. After you create the clone, you can then change details in the clone to meet the needs of new report. **Learn more.**



April 6, 2026

New bulk action: Update Optimization settings for multiple Ad Groups

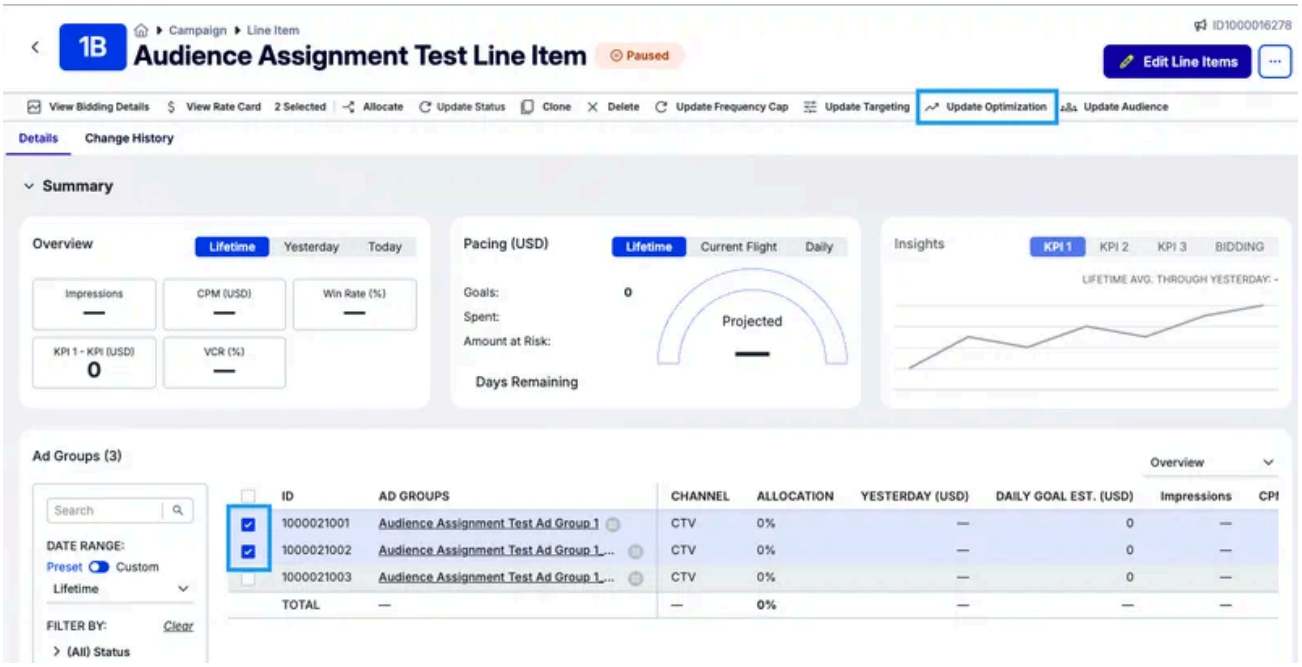
You can now apply optimization setting changes to multiple ad groups simultaneously, streamlining your campaign management workflow and ensuring consistency across your ad groups. Previously, updating these settings required editing each ad group one by one.

You'll find this bulk update option on the following pages:

- **Campaigns Details**: For the list of line items appearing there
- **Line Item Listing**: On the Campaigns page
- **Line Item Details Page**: For the list of ad groups appearing there

To bulk update ad group optimization:

1. Select one or more ad groups. Note that ad groups must be associated with the same advertiser to perform this bulk update.
2. Click **Update Optimization** to open the **Optimization** panel.
3. Click **Save** to apply your changes across all selected items.

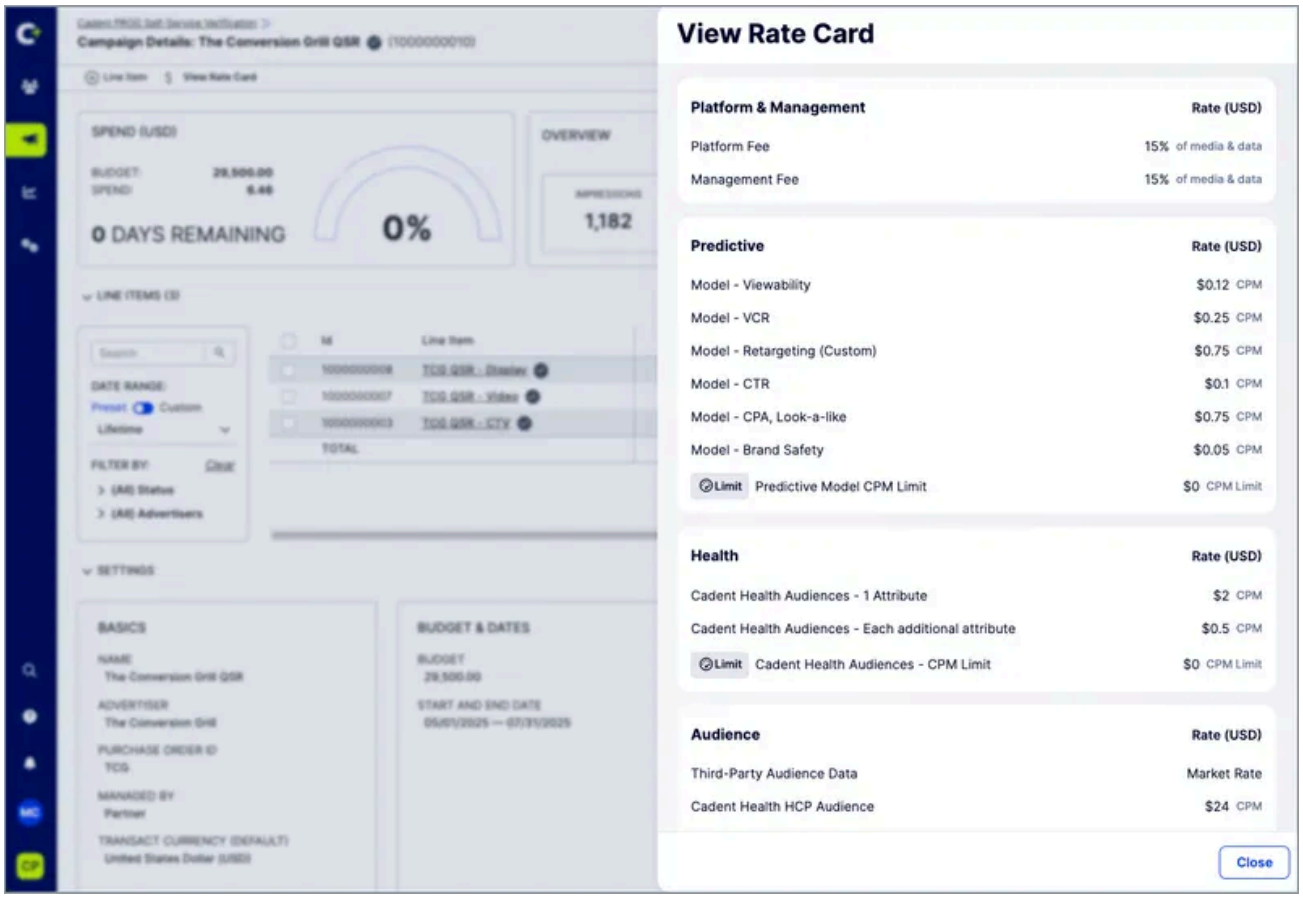


Easily reference your pricing with Rate Card Viewer

We're excited to announce the launch of the new Rate Card Viewer in Ad Manager. This new view provides quick, transparent access to your Rate Card, reducing time spent looking for your rates.

To access the Rate Card, click **View Rate Card** in one of the following screens:

- Campaign Details
- Line Item Details
- Ad Group Details



Rate card viewer

April 1, 2026

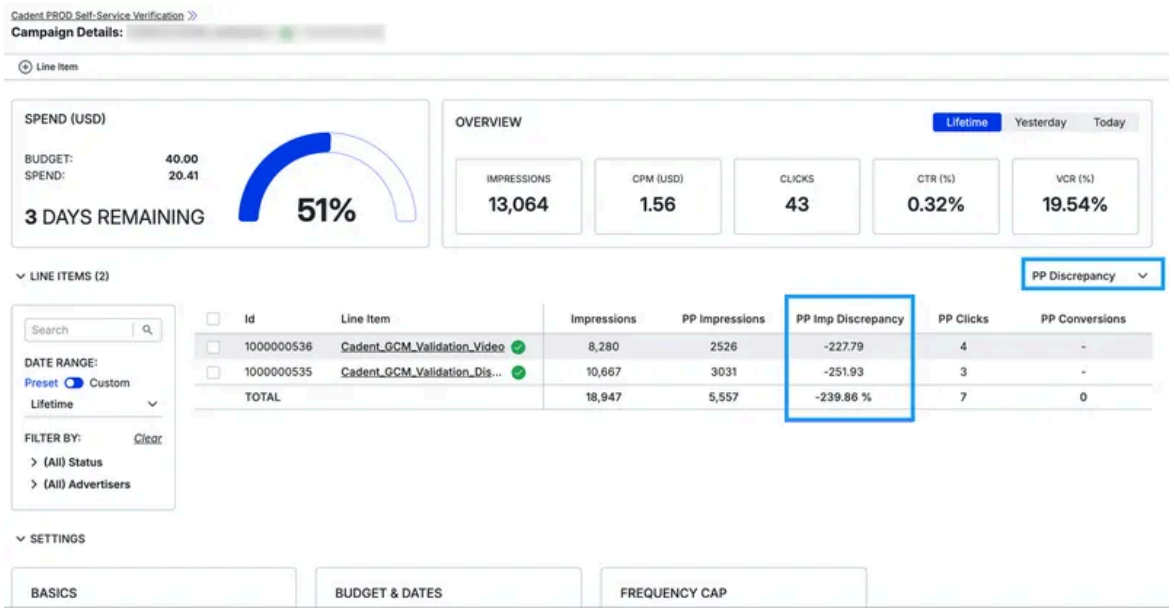
Ad Manager release notes

Resolved issue: Primary Partner Impression Discrepancy

The methodology used for calculating the difference between Cadent-tracked impressions and Primary Partner impressions (e.g., those tracked by an ad server like CM360) was displaying inaccurate numbers. We've resolved this issue, so now the **PP Imp Discrepancy** column is reflecting accurate numbers.

To view the PP Imp Discrepancy column:

1. Navigate to the **Campaign Details** or **Line Item Details** page.
2. Select **PP Discrepancy** from the view selector.



March 23, 2026

Ad Manager release notes

Report filters for Ad Hoc Reporting have been simplified

When creating a new report, you can now filter reports only by agency. No other filters are required. This makes it possible to produce more comprehensive reports than were previously possible. **Learn more.**

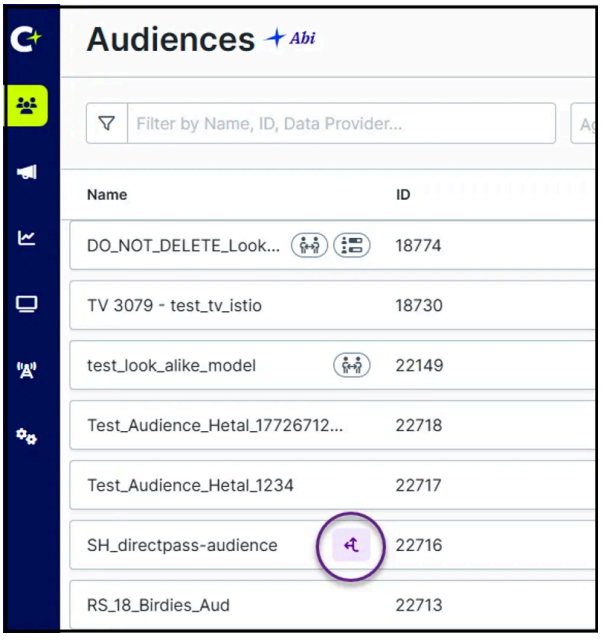
Audience Manager release notes

DirectPass icons added in Audience Manager

A new DirectPass icon now displays alongside DirectPass datasets in Audience Manager making them easier to spot. DirectPass uses a specialized dataset onboarding and deployment workflow that allows the direct ingestion and deployment of raw device identifiers (such as Device IDs and IP addresses) without first resolving them to Cadent's household graph. DirectPass datasets don't include the household level information.

DirectPass icons display in the following locations in Audience Manager:

- Audiences page
- Deployments page
- New Audience, My Datasets tab
- New Audience, Marketplace Datasets tab
- New Audience, Shared with Me tab


Audiences
Abi
Filter by Name, ID, Data Provider...
Name
ID
DO_NOT_DELETE_Look...
18774
TV 3079 - test_tv_istio
18730
test_look_alike_model
22149
Test_Audience_Hetal_17726712...
22718
Test_Audience_Hetal_1234
22717
SH_directpass-audience
22716
RS_18_Birdies_Aud
22713

DirectPass icon on the Audiences page

Audience tags display on the Audiences page

You can now see audience tags with audiences on the Audiences page. Tags provide the ability to create categories of audiences that share commonality. Although the tags do not display by default, you can add them to the page using the icon on the Audiences page.

The % Reach indicator on the New Audience page has been updated

On the**Total Reach** section of the New Audience page, the **% Reach** column has been renamed **Percentage Change** to improve clarity. This column displays the percentage change in total reach as you configure a new audience. It is updated in real time when you make changes to the audience you are building. Learn more

March 18, 2026

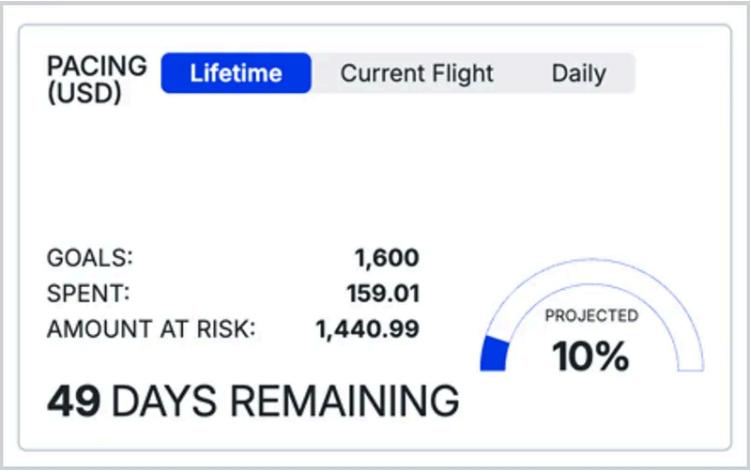
Ad Manager release notes

New 'Lifetime' view for line item pacing

We've added a new **Lifetime** timeframe to the line item Pacing tile. This allows you to see pacing metrics across all of a line item's flights, giving you a more holistic view of performance and helping you better understand and analyze the performance of your long-term campaigns.

Here's a breakdown of the metrics you'll see in this new view:

Metric	Description
Goal	This is the overall goal for the line item, depending on how your line item is pacing: • If pacing toward impressions, the Goal reflects Flight Impressions. • If pacing toward spend, the Goal reflects Flight Spend Cap.
Spent	The total amount spent through yesterday.
Amount at Risk	This shows the portion of the goal that's projected to be missed, giving you an at-a-glance view of pacing performance. Amount at Risk is calculated using this formula: (100% - Projected %) * Goal = Amount at Risk
Days Remaining	The number of days left in the current flight, not including today.



Simplified way to disable campaigns, line items, and ad groups

We've streamlined the process for disabling items in your account, making it faster and more intuitive to manage your campaigns.

Disabling campaigns and line items

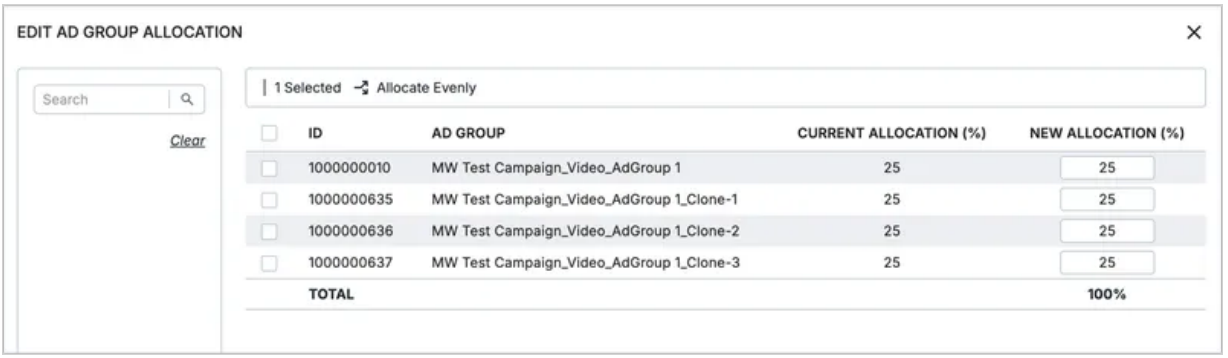
You can now directly set the status of an entire Campaign or Line Item to **Disabled** without changing its underlying budget setup. Previously, you had to go into each Ad Group individually to disable the allocation. This change makes pausing your campaigns a much quicker, one-step process.

Disabling ad groups

You can set an individual Ad Group to **Disabled**, but only if the parent Line Item's active budget allocation already totals 100%.

To check the budget allocation for a Line Item, simply follow these steps:

1. Go to the **Line Item Details** screen.
2. Select the **Ad Group** and click **Allocate**. This opens the Edit Ad Group Allocation panel.
3. Review the budget allocation details, and if necessary, make edits to bring the total allocation to 100%.



Save time with bulk pixel uploads

We're excited to announce a new feature that will make your workflow significantly more efficient. You can now create pixels in bulk—including universal tags, activities, and action activities—directly into Ad Manager, saving you from the manual effort of creating them one by one.

How to Upload Pixels in Bulk

Follow these steps to get started:

1. Navigate to the **Pixels** screen from the main menu.
2. Click the **+Pixels** button and select the **Bulk** option. This will open the Add Pixels in Bulk panel.

3. Select the **Advertiser** you want to associate the pixels with.
4. Download the **sample file** provided. This template ensures your data is formatted correctly for a successful upload. Add your pixel information to this file and save it to your computer. If you're trying to make any new activities for an existing Universal Tag, you should download a new template from the panel to ensure your existing Universal Tags are pre-populated in the dropdown list on the Activities tab of the template.
5. Drag and drop your completed file into the upload area, or click to browse for it on your computer.
6. Click **Add**.

Once the upload is complete, you will be automatically returned to the main Pixels screen where you'll see a success message confirming your new pixels have been added.

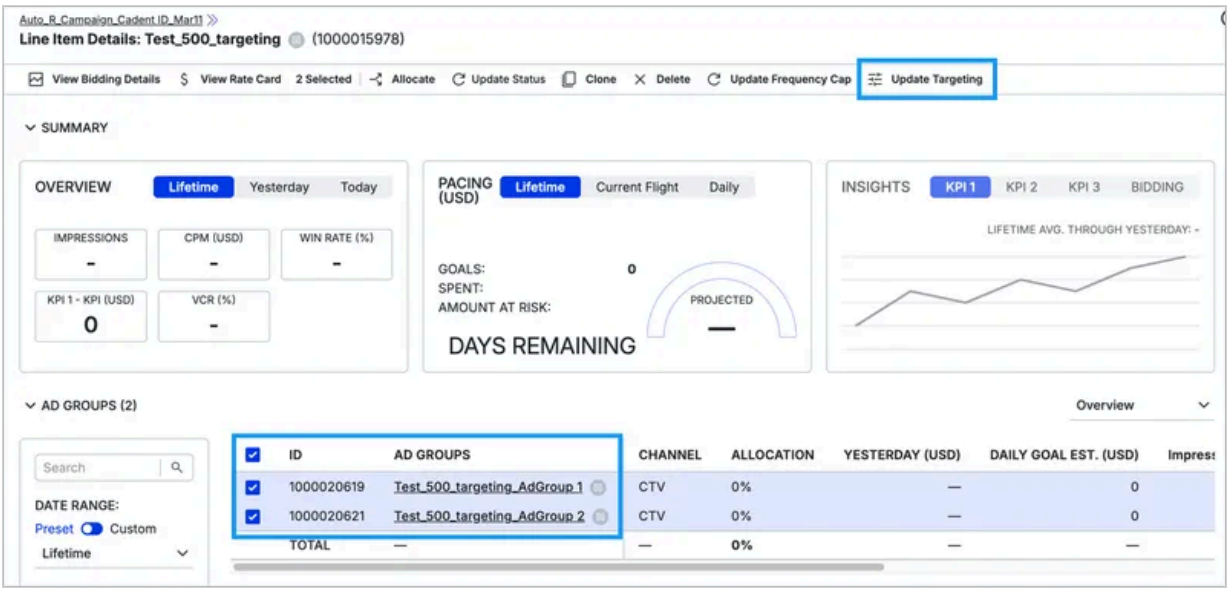
Streamline your targeting with bulk actions

You can now update **Custom Categories**, **Supply Categories**, and **Network & Channel** targeting for multiple ad groups at the same time.

> **Key Details to Remember**
>
> - **Custom Categories:** You can select from pre-made custom categories during a bulk update. New categories must be created beforehand.
> - **Network & Channel:** This targeting option is specifically for CTV.
> - **Supply Categories:** These are Jounce-defined supply categories.

How it works:

1. Navigate to the Line Items Details screen and select the ad group(s) you want to update, and then click **Update Targeting**. This opens the Add Targeting panel.



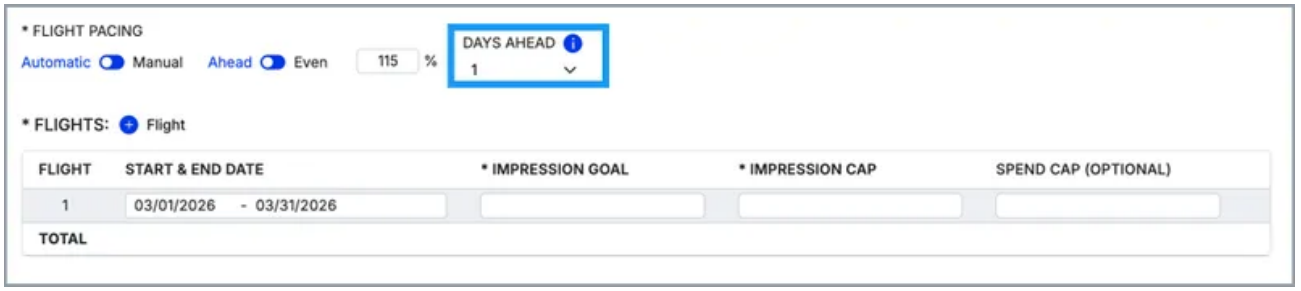
2. In the **Add Targeting** panel, make your new targeting selections and close the panel.
3. You're all set! The new targeting options will be reflected in the **Ad Group Details** screen for each Ad Group you updated.

February 19, 2026

Ad Manager release notes

Line item update: Days Ahead setting

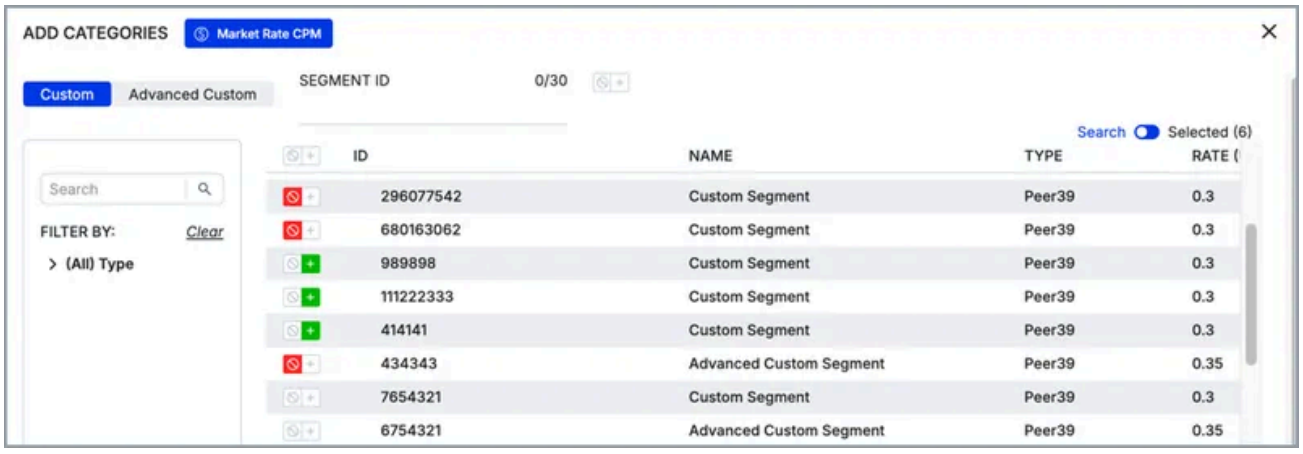
You now have the ability to indicate whether a line item should aim to deliver in full a certain numbers of days prior to the end date.



Targeting update: Categories targeting

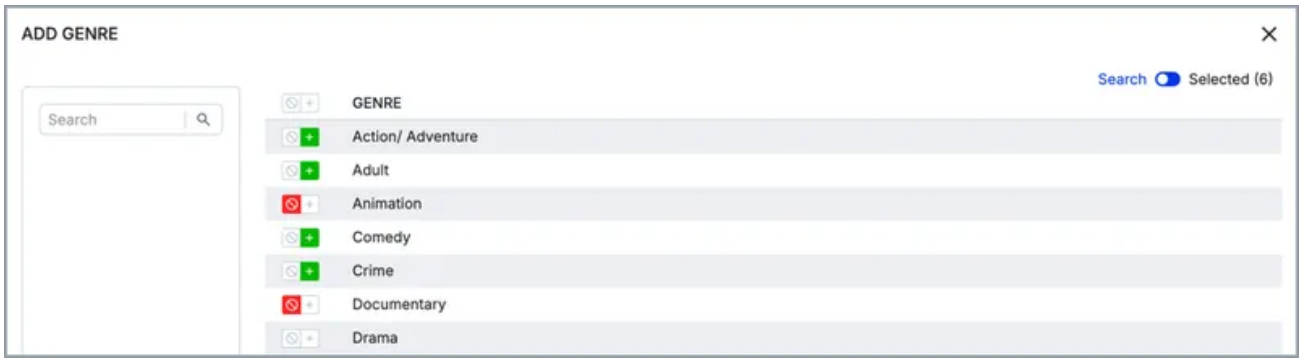
You can now target IAB or Peer39 content categories, including custom and advanced Peer39 segment creation and selection, to ensure your ads appear on relevant content.

You can target specific IAB categories or Peer39 segments, or you can exclude categories to prevent your ads from appearing on undesirable content.



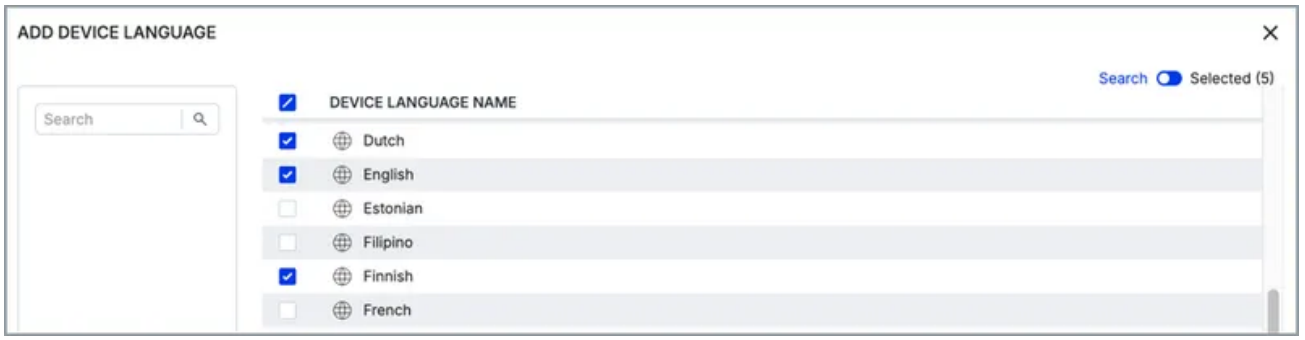
Targeting update: Genre targeting

Genre targeting allows you to align your campaigns with specific types of CTV video content. This feature enables you to target or exclude content based on the genre—for example, Comedy, Sports, Drama, Documentaries, and True Crime. You can either include specific genres to target or exclude genres you want to avoid.



Targeting update: Device Language targeting

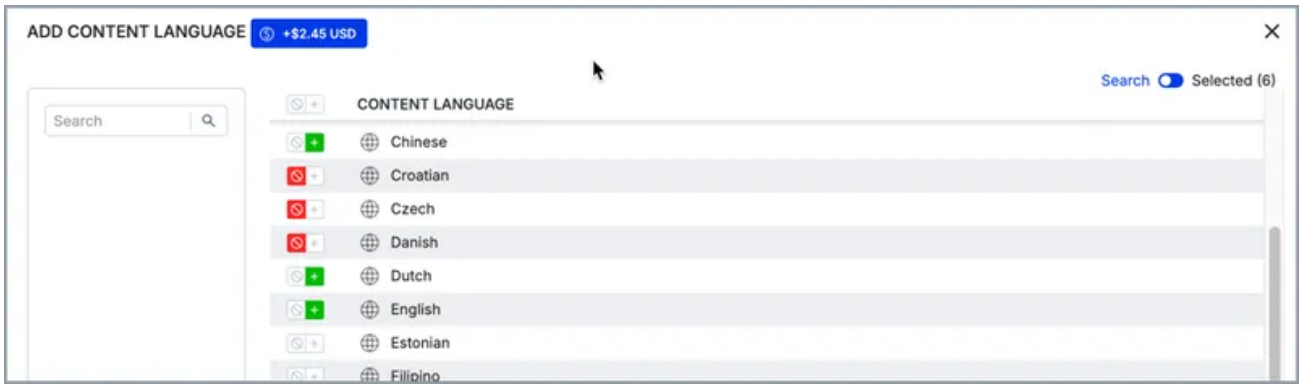
Device Language targeting lets you deliver ads specifically to users whose devices are set to a particular language. It's a straightforward way to ensure your message is relevant and understood by your target audience, especially in multilingual regions or global campaigns.



Targeting update: Content Language targeting

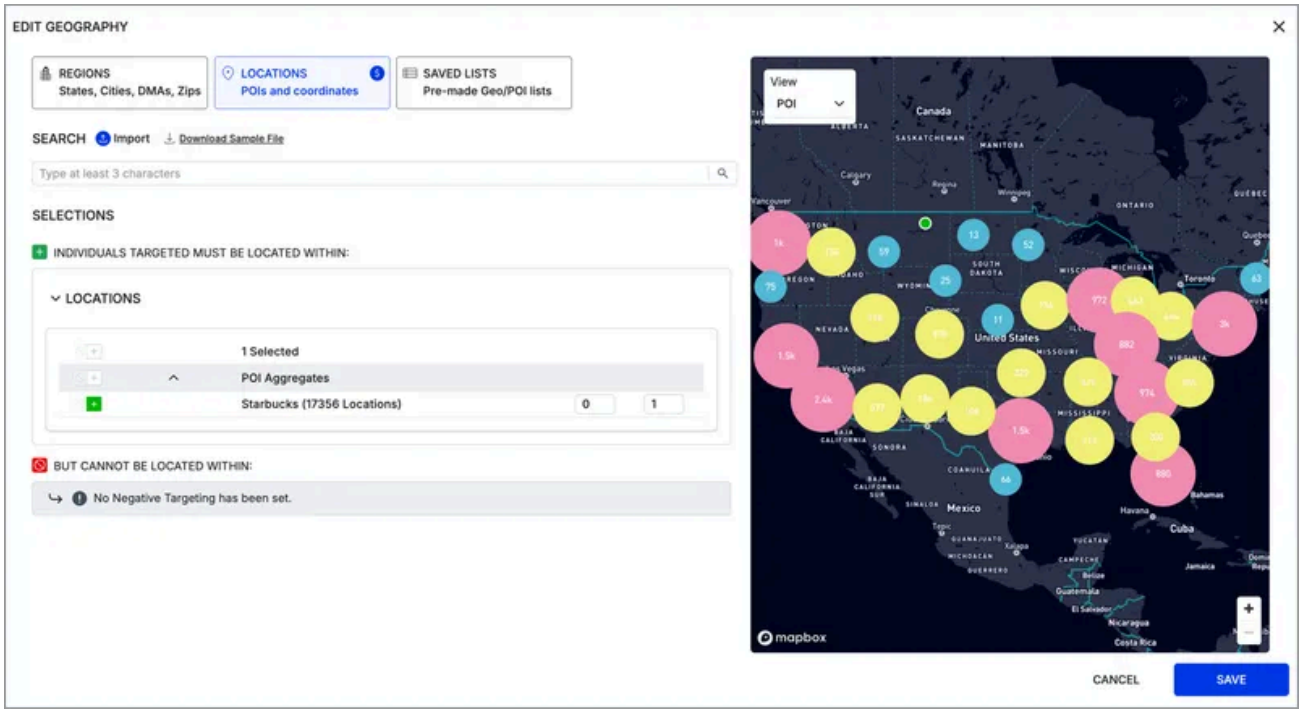
This method allows you to target viewers based on the primary language of the CTV content itself. For example, you can choose to have your ads appear only during Spanish-language programs, regardless of the viewer's device settings.

You can choose to either target or exclude language.



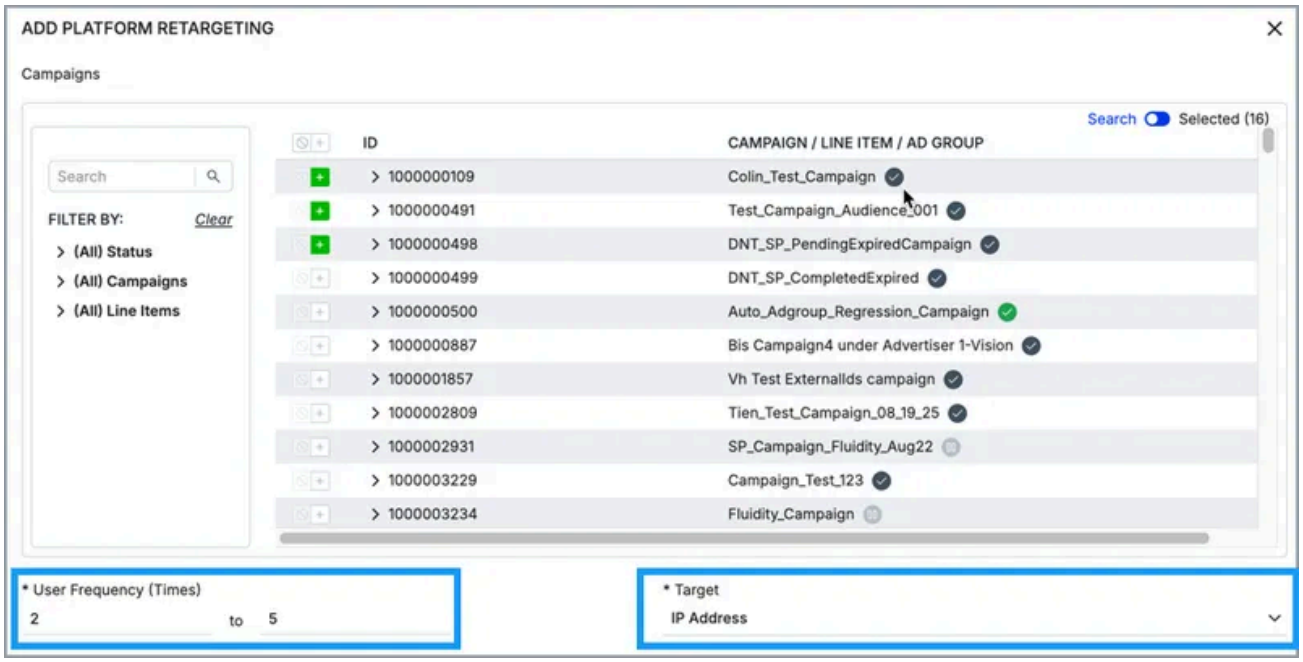
Targeting update: Geo POI targeting

You can target POIs via Geo targeting, which are presented to the user as an aggregate vs single point selection. For example, in one click you can target 4,000 Starbucks vs. having to select each one at a time.



Targeting update: Platform retargeting

Platform retargeting allows you to target users who have previously exposed to your campaigns.



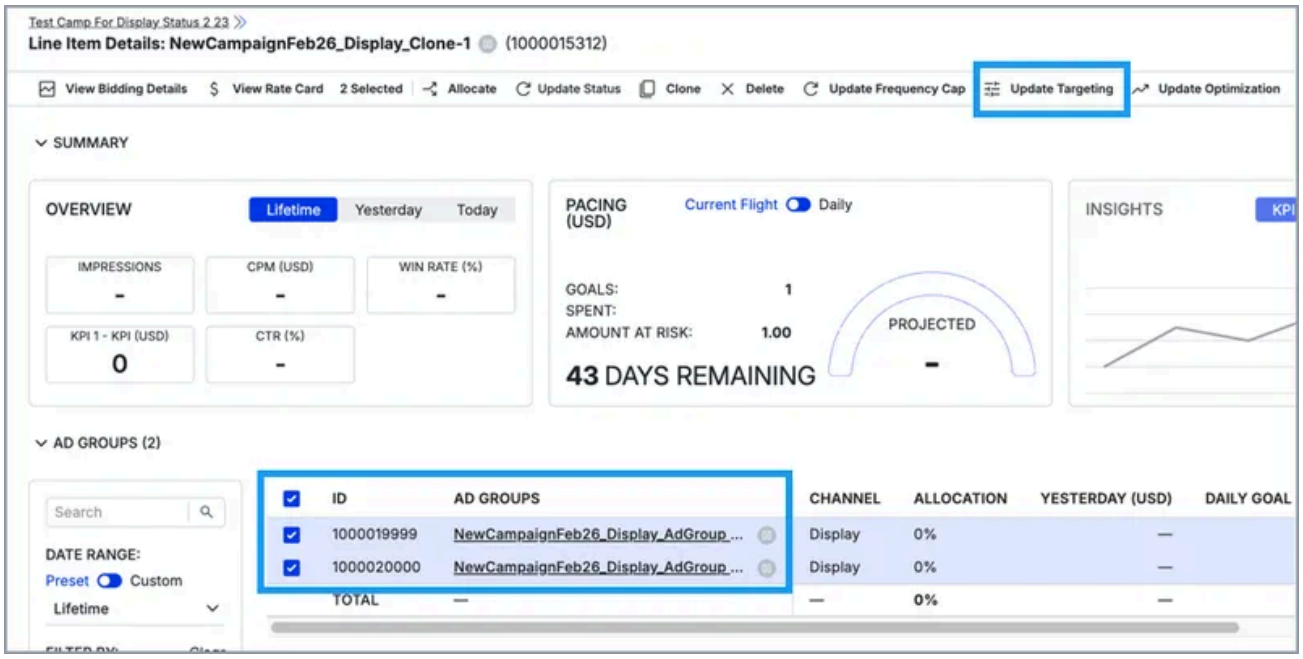
- You can choose to either positively target (include) or negatively target (suppress) the selected audience.
- User Frequency:This allows you to set a minimum and maximum range for how many times a person has been served your ad. For example, if you set a minimum of 2 and a maximum of 5, the system will only target users who have seen the ad at least 2 times but no more than 5 times.
- Target: This is the identifier used to count the frequency. You can select to track users by their IP Address, a browser Cookie, or All (which uses both methods).

Targeting update: Bulk actions

You can now select multiple ad groups in the line item details view to apply bulk targeting actions (see the screenshot below). The selections you make will override the previous selections.

The targeting options you can bulk update are:

- App/Sites list
- Packages
- Geography
- Deals
- Keywords



Reporting release notes

Fewer filters are required when creating an ad hoc report

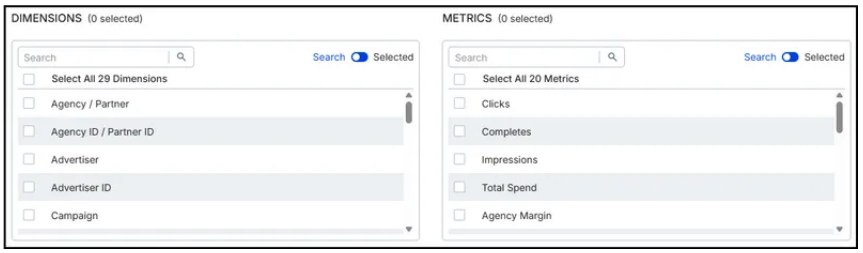
We have reduced the number of filters you need to apply when creating an Ad Hoc report. Now, you are only required to select an agency. This makes it possible to create broader reports than were possible in the past. Learn more about **Creating ad hoc reports**.

New reporting dimension added to Performance Insights templates

To provide you with a way to see more detail on your creatives in reports, a new reporting dimension, **Creative Type**, has been added. A creative type defines how a user interacts with a digital ad. For example, **video**, **display**, and **banner** are creative types that apply to online ads. The creative type dimension is being made available on the following report templates:

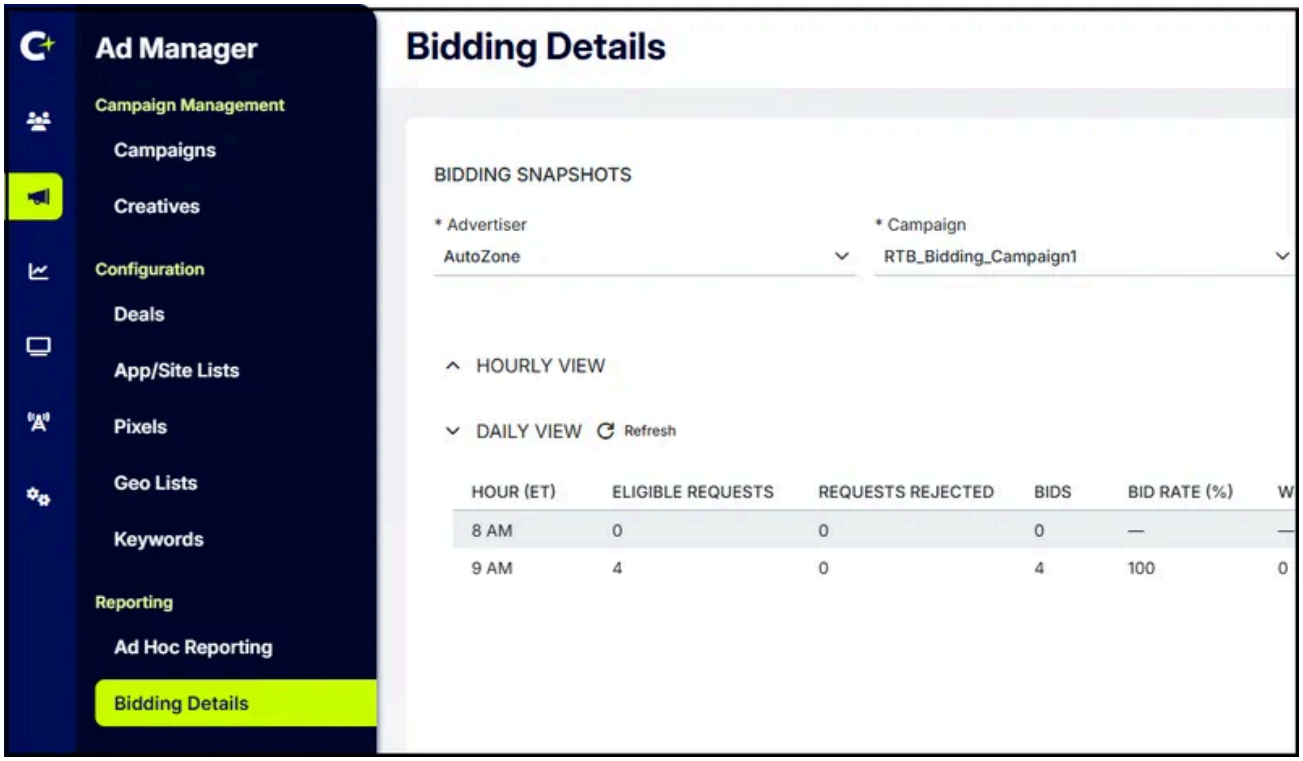
- Campaign metrics
- CTV metrics
- Video metrics
- Geo targeting metrics
- Reach and frequency
- HCP metrics

Learn more about **Performance Insights Templates**.



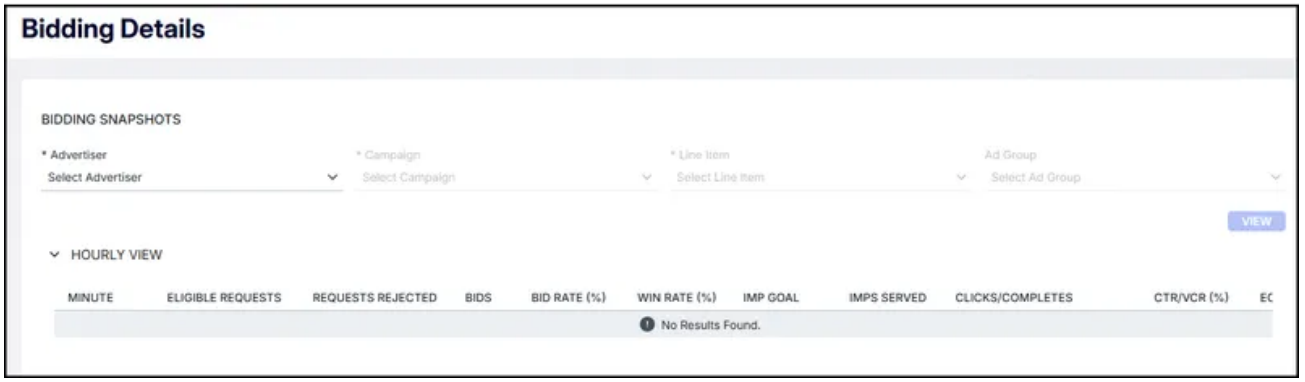
Bidding Snapshot renamed Bidding Details

When you access the Bidding Snapshot report, you will notice that it has been renamed **Bidding Details**. The new name displays on the left-side navigation bar and on the top of the page in Ad Manager. It is the same report as before but with two new additions described below.



Bidding Details report adds an hourly view

In addition to the Daily and Troubleshooting views, you now have the option to select an **Hourly View** of line items and ad group details. The hourly view displays the current hour's activity, and it can be refreshed to show the latest updates.



New fields display on Bidding Details report

The following new fields have been added to the Bidding Details report results for hourly and daily report results:

- **Spend Goal** - The monetary target an advertiser or media buyer aims to reach within the timeframe of a line item or ad group.
- **Total Spend** - The amount of money your campaign or line item has spent from the budget to win impressions.

February 4, 2026

Audience Manager release notes

DirectPass Data Ingestion

DirectPass uses a specialized dataset onboarding type and deployment workflow that allows the direct ingestion and deployment of raw device identifiers, such as Device IDs and IP addresses, without first resolving them to Cadent's household graph. DirectPass datasets don't include the household level information.

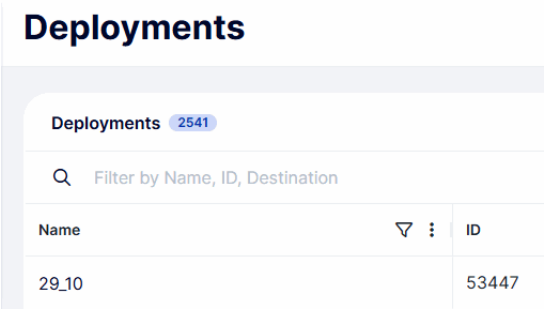
DirectPass enables specialized workflows that require the use of raw, unaltered identifiers. You can use DirectPass under the following conditions:

- Targeting for regions outside the United States, such as Canadian segments for use in AdManager.
- Passing raw identifiers directly to destination partners that have their own identity resolution systems.

Note: You must have DirectPass enabled for your organization to perform this type of data onboarding. If interested, contact your Account Manager.

ID filter added to Deployments page

The filter on the Deployments page has added the option to search by Deployment ID.
Deployments page details



January 23, 2026

Health Audience release note

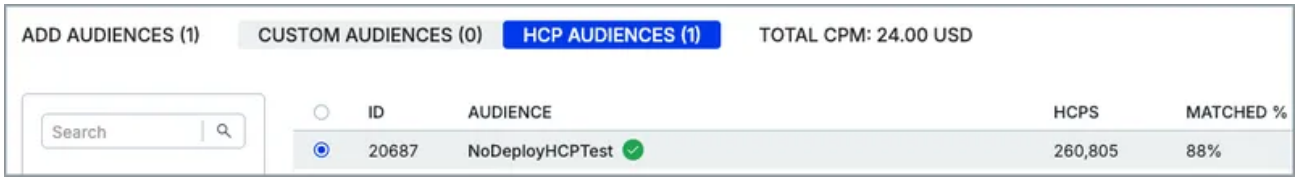
Assign an HCP audience to an ad group

What's an HCP audience?

A Healthcare Professional (HCP) audience is a target group consisting of healthcare professionals such as doctors, nurses, pharmacists, and dentists. When you build and save an HCP audience in the **Health Audience builder**, it's automatically available for assignment in the UI—as long as the audience and ad group are both associated with the same advertiser.

How to assign an HCP to an ad group

You can now assign an HCP audience to your ad group during the ad group setup. See **Add an ad group** for instructions.



Audience assignment screen in Ad Manager

13. CrossRiver test page

asopdkjasoifnsdijfnsadjlfnsad]

> asdsadsadsadsa

14. CrossRiver test page 2

Relevant webhooks

For this endpoint be sure you have **Accounts, cards and payment events** for this webhook:

- **Rtp.Payment.Received**
- **Rtp.Payment.Expired**
- **Rtp.Payment.Rejected**

API reference

Initiates a request for payment (RfP)
Allows the recipient to authorize and trigger an instant payment credit transfer. RfPs can be initiated from:

- Master account
- Deposit account (DDA)
- Associated subledger

Endpoint: /rtp/v1/payments/payment-request

POST https://sandbox.crbcos.com/rtp/v1/payments/payment-request

Request | Response

BODY PARAMETERS

accountNumber String required

Account from which the RfP is originated

amount Integer required

Requested payment amount in positive integeral cents. For example, represent $1.00 as 100.

purpose String optional

Internal CR field used to include a short description or purpose for the payment request. The RDFI does not see this field. 50 characters maximum.

requestedExecutionDate String required

Date when the payment request needs to be processed by the Debtor. Formatted as yyyy-mm-ddThh:mm:ss[.mmm].

expiryDate String required

Expiration date and time of the RfP. If no time is specified, it expires at 12:00 AM on the given date. If a time is provided, it expires within one hour after the specified time on the same date. Formatted as yyyy-mm-ddThh:mm:ss[.mmm].

clientIdentifier String optional

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes. 50 characters maximum.

remittanceData `String` `optional`

A description of the payment request, meaning, the remittance advice. Use this field for any remittance information that doesn't have its own field. 140 characters maximum.

remittanceLocation ▶ `Object` `optional`

Remittance location information

debtor ▶ `Object` `required`

Debtor Personal/Business information. Debtor is the individual receiving the Request for Payment. Please note: For the debtor, name and one additional element like [birthCity, birthCountry, birthdate], [orgLegalEntityId/LEI], OR [addressStreetName, addressCity, addressState, addressPostalCode, and addressCountryare] required as per RfP business rules. edit

ultimateDebtor ▶ `Object` `optional`

Ultimate Debtor Business/Individual Information. The final party receiving the RfP and responsible for initiating the credit transfer upon authorization.

creditor ▶ `Object` `optional`

Creditor Business/Individual Information. The recipient of the payment, typically the party at CRB initiating the RfP. Depending on the configuration, this information may be populated automatically.

ultimateCreditor ▶ `Object` `optional`

Ultimate Creditor Business/Individual Information. The final recipient of the payment. Depending on the configuration, this information may be populated automatically.

useCase `String` `required`

Use one of the following values for the useCase field:

-BusinessToBusiness
-ConsumerBillPay
-AccountToAccount
-DownPaymentOrFinalPayment
-ConsumerToGovernment

industryCategory `String` `optional`

Use one of the following values for the industryCategory field (optional when useCase is BusinessToBusiness or ConsumerToGovernment):

-HomeUtilities
-TelecomUtilities
-Mortgage
-Rent
-InsurancePremiums
-AutoLoans
-PersonalLoans
-CreditCardPayment
-MediaSubscriptions
-Memberships
-FinancialInstitution
-BrokerOrDealer
-Medical
-OnlineGaming
-RealEstate
-Government

senderID `String` `required`

Use the senderId field (optional when useCase is BusinessToBusiness or ConsumerToGovernment), provided by Cross River and generated by The Clearing House (TCH).

Valid senderID's for Sandbox Testing:
-100000000001
-100000000002
-100000000003
-100000000004
-100000000005

amountModAllowed `Boolean` `optional`

Indicates whether the recipient is allowed to modify the requested amount in response to an RfP. If the field is not provided, the system defaults to true.
If true, the recipient may respond with a different amount
If false, the recipient must match the original amount; any mismatch will cause the payment to be rejected.

```
curl --location 'https://sandbox.crbcos.com/rtp/v1/payments/payment-request' \
--header 'Content-Type: application/json' \
--data '{
    "accountNumber": "2564791479",
    "amount": 20000,
    "discount": 1000,
    "debtor": {
        "routingNumber": "071212128",
        "accountNumber": "2758463448",
        "name": "Terry Tester",
        "birthCity": "Pine Brook",
        "birthCountry": "US",
        "birthDate": "1996-08-09",
        "addressStreetName": "Linwood Ave",
        "addressBuildingNumber": "2115",
        "addressCity": "Pine Brook",
        "addressState": "NJ",
        "addressPostalCode": "07024",
        "addressCountry": "US"
    },
    "creditor": {
        "accountNumber": "2564791479",
        "name": "the creditor"
    },
    "purpose": "Test Payment 2",
    "clientIdentifier": "123456",
    "requestedExecutionDate": "2025-08-10",
    "expiryDate": "2025-08-16",
    "senderId": "100000001",
    "useCase": "BusinessToBusiness",
    "industryCategory": "OnlineGaming"
}'
```

```
{
    "id": "8ee74810-45fe-47ee-9569-b30500b8d6ae",
    "originalPaymentId": "8ee74810-45fe-47ee-9569-b30500b8d6ae",
    "referenceId": "R174OI808VYXJ",
    "accountNumber": "2955057589",
    "clientIdentifier": "123456",
    "amount": 19000,
    "direction": "Outbound",
    "status": "Created",
    "paymentType": "RequestForPayment",
    "source": "Api",
    "transactionAccountContext": "NotSubmitted",
    "debtor": {
        "routingNumber": "011000138",
        "accountNumber": "2758463448",
        "name": "Terry Tester",
        "birthCity": "Pine Brook",
        "birthCountry": "US",
        "birthDate": "1996-08-09",
        "addressStreetName": "Linwood Ave",
        "addressBuildingNumber": "2115",
        "addressCity": "Pine Brook",
        "addressState": "NJ",
        "addressPostalCode": "07024",
        "addressCountry": "US"
    },
    "creditor": {
        "routingNumber": "021214891",
        "accountNumber": "2955057589",
        "name": "Dim Mak",
        "addressStreetName": "400 Kelby St",
        "addressCity": "Fort Lee",
        "addressState": "NJ",
        "addressPostalCode": "07024",
        "addressCountry": "US"
    },
    "network": {
        "clearingSystemCode": "TCH",
        "currency": "USD",
        "endToEndId": "6909c29416f0453fbdfdb30500b8d6ae",
        "instructionId": "20250623021214273T1B19TS07136537534",
        "interbankSettlementAmount": 19000,
        "interbankSettlementDate": "2025-06-23",
```

```
"messageId": "M20250623021214273T1BDQK15672530466",
"numberOfTransactions": 1,
"settlementMethod": "CLRG",
"businessMessageId": "B20250623021214273T1BPRK46725920502",
"toParticipantId": "990000001S1",
"fromParticipantId": "021214273T1",
"instructingRoutingNumber": "021214891",
"instructedRoutingNumber": "011000138",
"headerCreationDate": "2025-06-23T07:12:58.6058948-04:00",
"messageCreationDateTime": "2025-06-23T07:12:58.6058948-04:00",
"createdAt": "2025-06-23T07:12:58.6058948-04:00"
```

14.1. Untitled

15. rețete

sdsadasdsa		sadasdasd

asasdasdas		

Get Cakes

Get a cake by its ID

GET https://api.cakes.com

Request

BODY PARAMETERS

id String required

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");

var raw = JSON.stringify({
   "id": "string"
});

var requestOptions = {
   method: 'GET',
   headers: myHeaders,
   body: raw,
   redirect: 'follow'
};

fetch("https://api.cakes.com", requestOptions)
   .then(response => response.text())
   .then(result => console.log(result))
   .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```

16. Test

klnklk

asdasdasda

17. Ashbyk img alt text dissapear

Learn how to accelerate candidate reviews by using AI to evaluate applicants based on custom criteria. This feature helps your team identify top talent faster by automating the initial resume screening process directly within Ashby.

Permissions & Access

- **Plans:** Available on Foundations, Legacy Plus, Plus, and Enterprise.
- **Permissions:** You must be an Org Admin to enable the feature. To edit job-level criteria, you need Elevated Access - Admin or Elevated Access - Admin Private for that specific role.
- This is an **opt-in feature** that must be turned on. You can opt-in by navigating to **Admin > Organization Setup > Opt-In Features** and enabling **AI-Assisted Application Review.**

Step 1: Enable AI-assisted application review

1. Navigate to **Admin > Organization Setup > Opt-In Features.**
2. If **Enable AI Features** is not enabled, turn it on it now. You will need to accept Ashby's terms for AI Features to continue.
3. Then, enable **AI-Assisted Application Review** from the list of available AI feature tiles.

Step 2: Define your review criteria

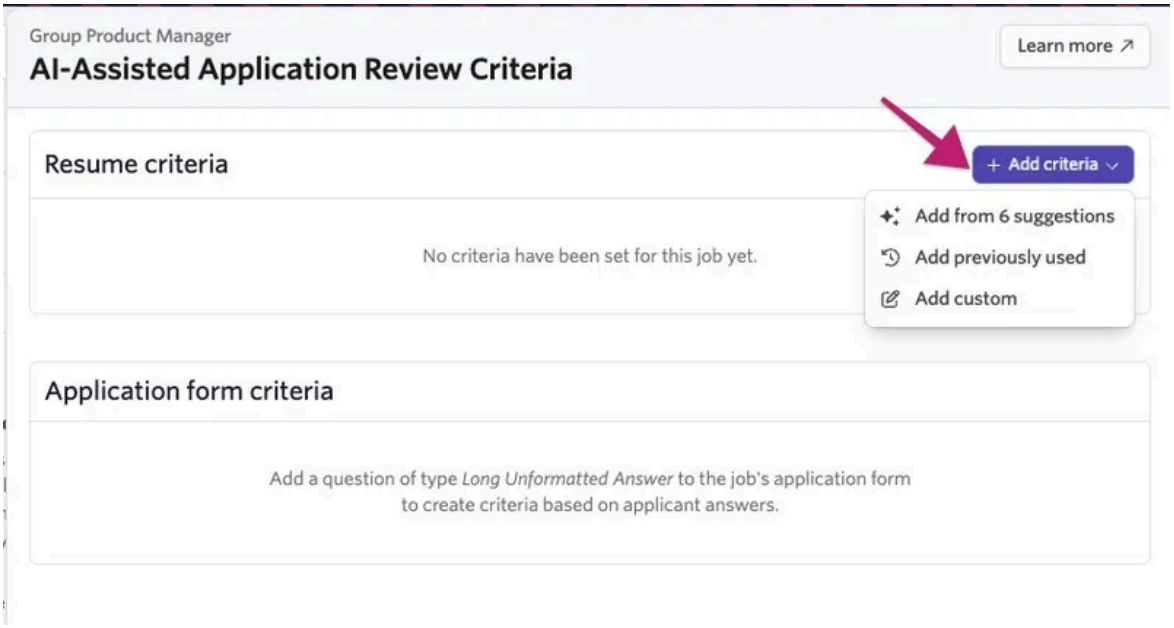
Criteria that you add to the job defines what the AI assesses. You can define resume criteria that the AI looks for within a candidate's resume as well as whether the candidate answered specific application form questions.

To define your criteria, navigate to **Jobs > Jobs** and select a job. In the job settings, navigate to the **AI Features** tab and select `Set up criteria`.

Resume criteria

In the Resume criteria section, you can add criteria in multiple ways: add criteria suggested by AI based on the job posting, add previously used criteria from other jobs, or add custom criteria that you create for this role specifically.

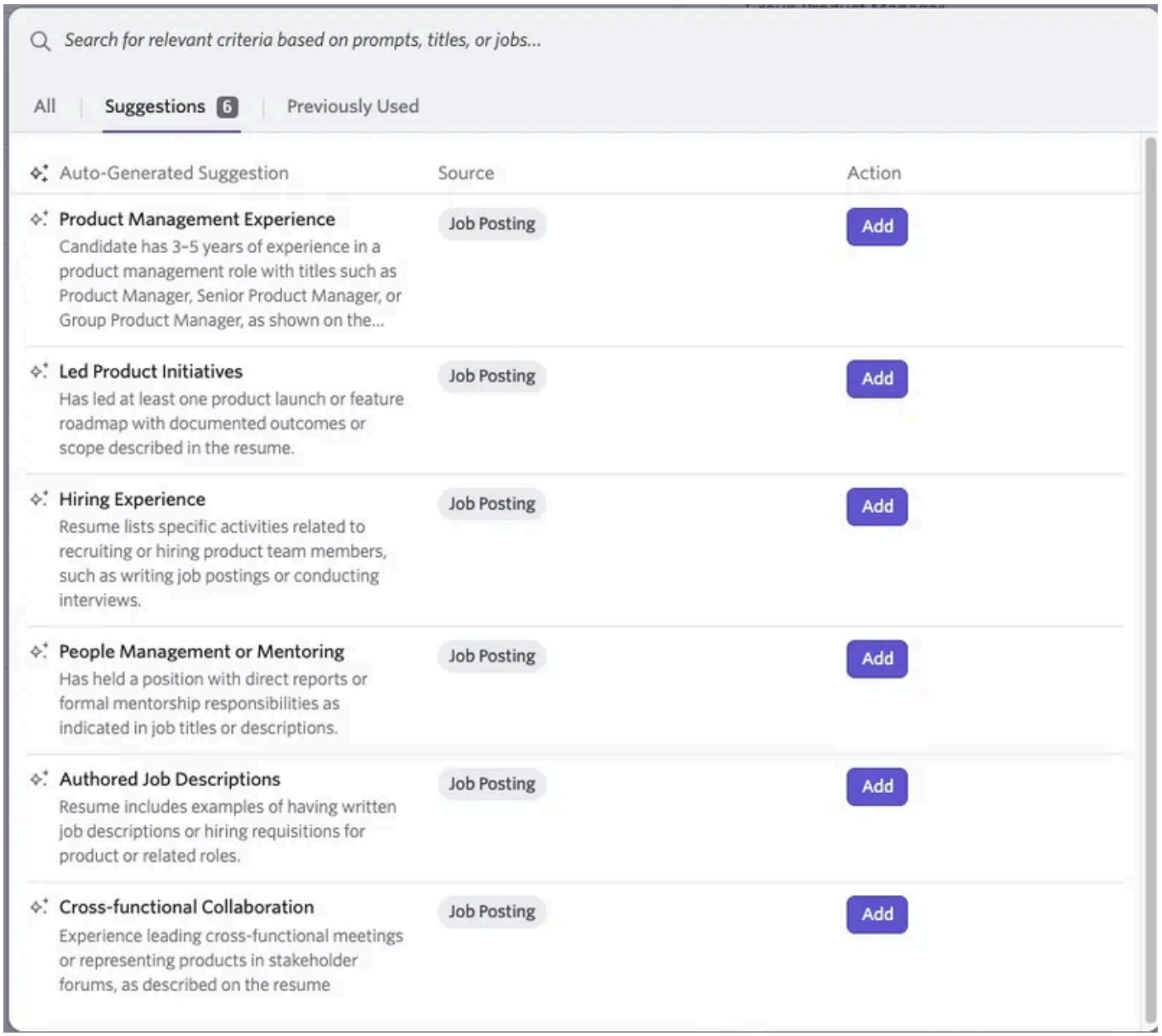
To start adding criteria select `+ Add criteria` under the **Resume criteria** section.



Add suggested criteria or previously-used criteria

Select `Add from # suggestions` for suggestions generated by AI based on your job posting. You can also reuse criteria from other roles you have permission to access by selecting `Add previously used`.

To add a suggestion, select `Add` in the **Action** column.



Creating effective custom criteria

Selecting `Add custom` instead of the suggested or previously-used options allows you to create your own criteria. You'll need to define the following:

- **Short title:** The name of your criterion. This appears in job settings, the application review page, and as a filter name.
- **Prompt:** The instructions used to evaluate applicants. Precise and clear prompts result in better AI performance.

Tips on criteria effectiveness

> Write your criteria as if you were instructing a junior recruiter. Ensure the prompt is phrased so that any reader would reach the same conclusion.

- **Stand-alone:** Criteria should be understandable without extra context. Instead of "Can assimilate feedback," use "Effectively incorporates feedback related to personal job performance."
- **Singular:** Focus on one skill per requirement. Instead of "Experience in Python and AWS," create two separate criteria for each.
- **Clear:** Avoid vague terms like "technical background." Specify the exact expertise required.
- **Specific:** Use measurable requirements, such as years of experience or specific previous job titles.
- **Objectively verifiable:** Ensure the requirement can be proven by looking at a resume. Instead of "has good taste," use "has led design for consumer SaaS products."

Application form criteria

You can evaluate whether a candidate answered long-form application questions in the **Application Form Criteria** section.

Toggle the `Did the Application answer the question?` option for individual questions. These are added as criteria for the review process.

> This check only determines if the question was answered. Blank or unrelated answers are flagged as not meeting the criteria. This is only available for long, unformatted text fields.

Step 3: Validating and reviewing criteria

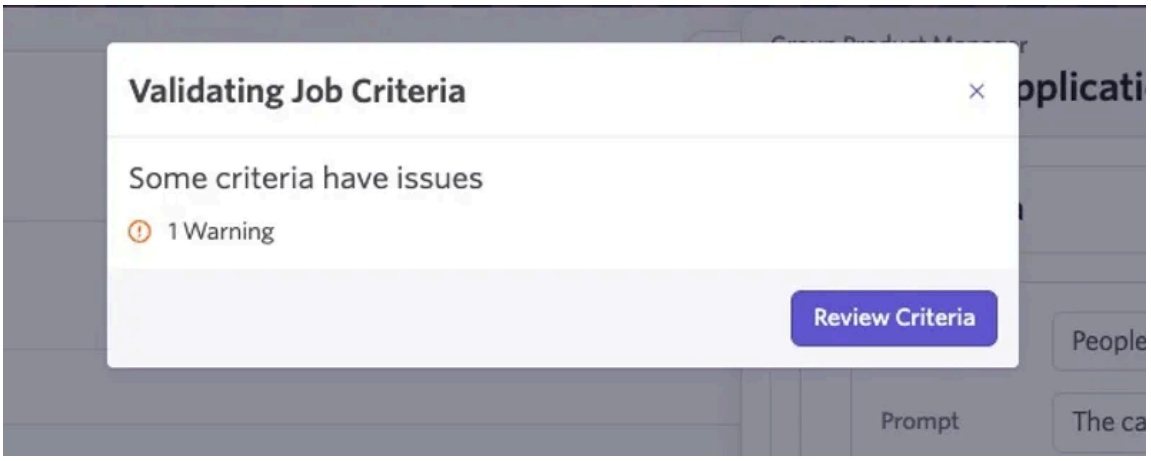
Once you have set up your criteria, it must be validated. Ashby checks your prompts on two dimensions:

- **Clarity and objectivity** to ensure that the criteria is clear and objective which means it can be appropriately evaluated by the AI.
- **Equal Employment Opportunity (EEO) violations:** the AI also scans for implicit or explicit EEO violations and flags them.

Validating criteria

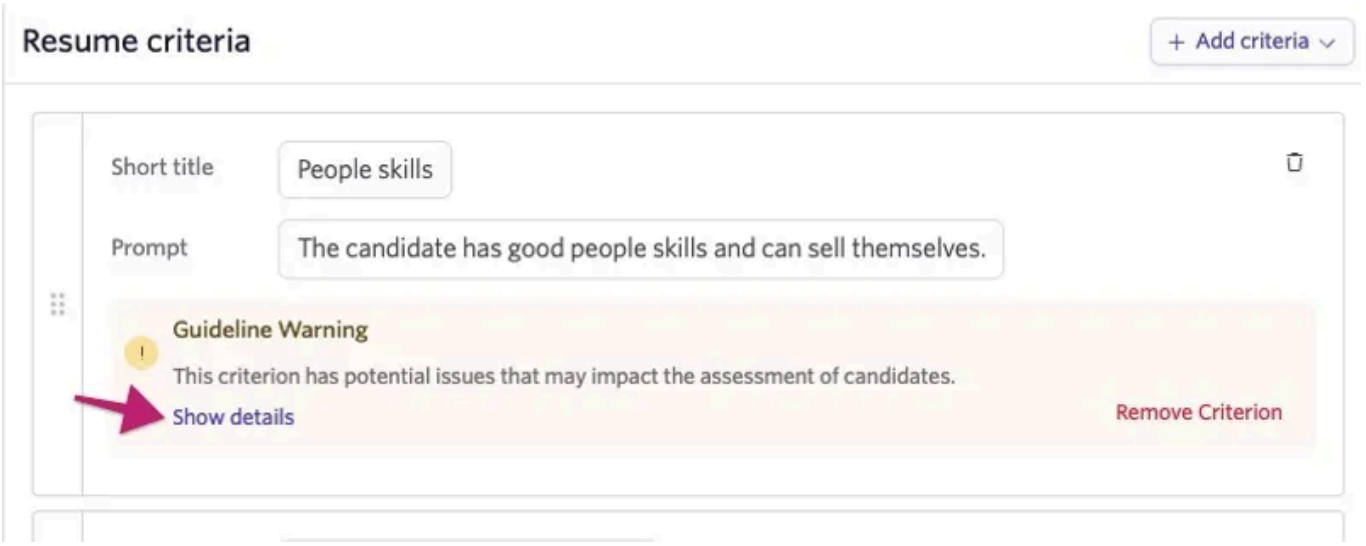
To validate your criteria, select `Validate`. If all criteria pass validation, you'll see a pop-up indicating that. Hit `Close` and proceed to the next section on Saving criteria and evaluating candidates.

If the system identifies issues, select `Review Criteria` on the pop-up to view warnings.



Reviewing flagged criteria

The flagged criteria will be identified with a warning. Select `Show details` under any warning to review why the criterion was flagged.



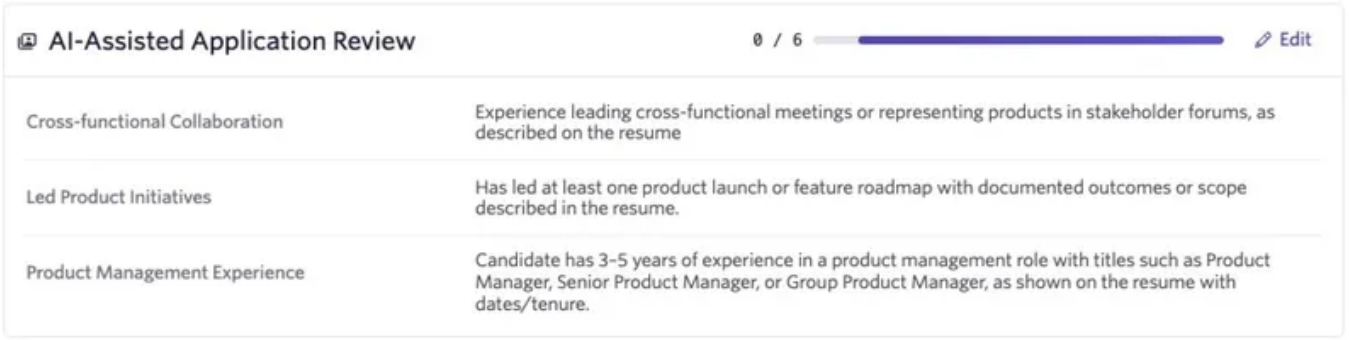
We recommend to either update the criterion until it is no longer flagged, or remove the criterion all together before proceeding.

> You can proceed with flagged criteria but it is not recommended.

Step 4: Saving criteria and evaluating candidates

After validation is successful, select `Save and Evaluate all active candidates` to begin the initial evaluation. Depending on the number of candidates, this can take a few seconds to a few minutes.

You'll see a loading screen with the total number of candidates being evaluated and a progress status bar.



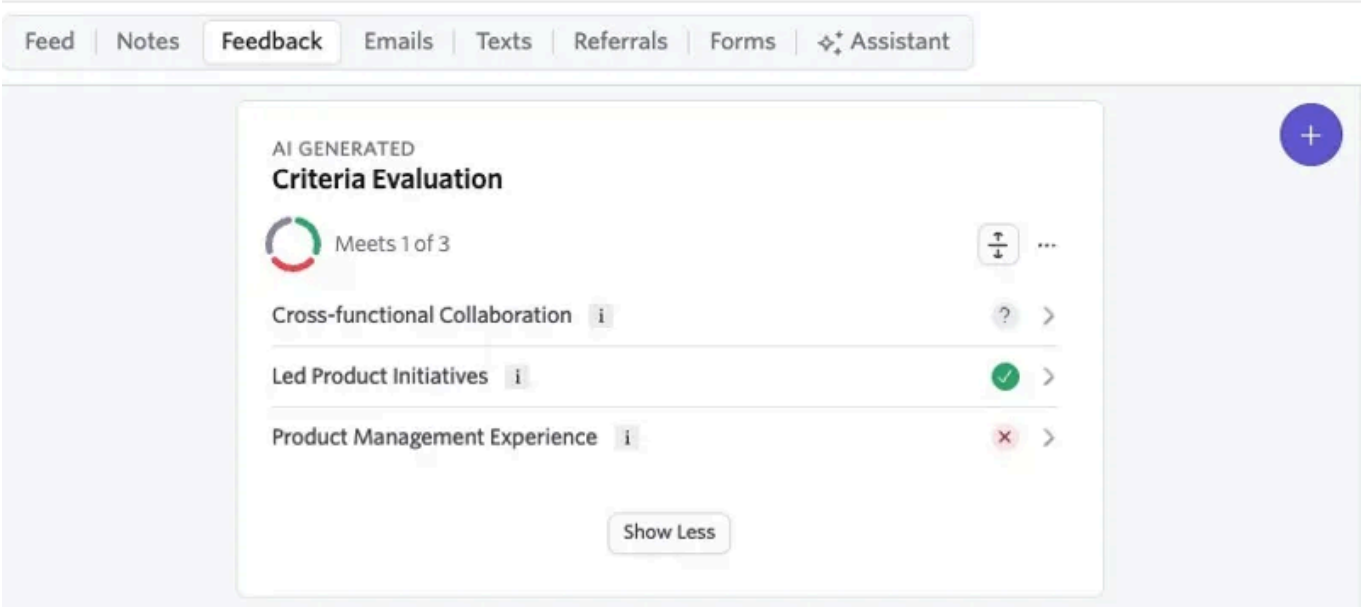
If you are updating existing criteria, select `Save and re-evaluate all active Job Considerations` once you've made your updates.

Review candidate evaluations

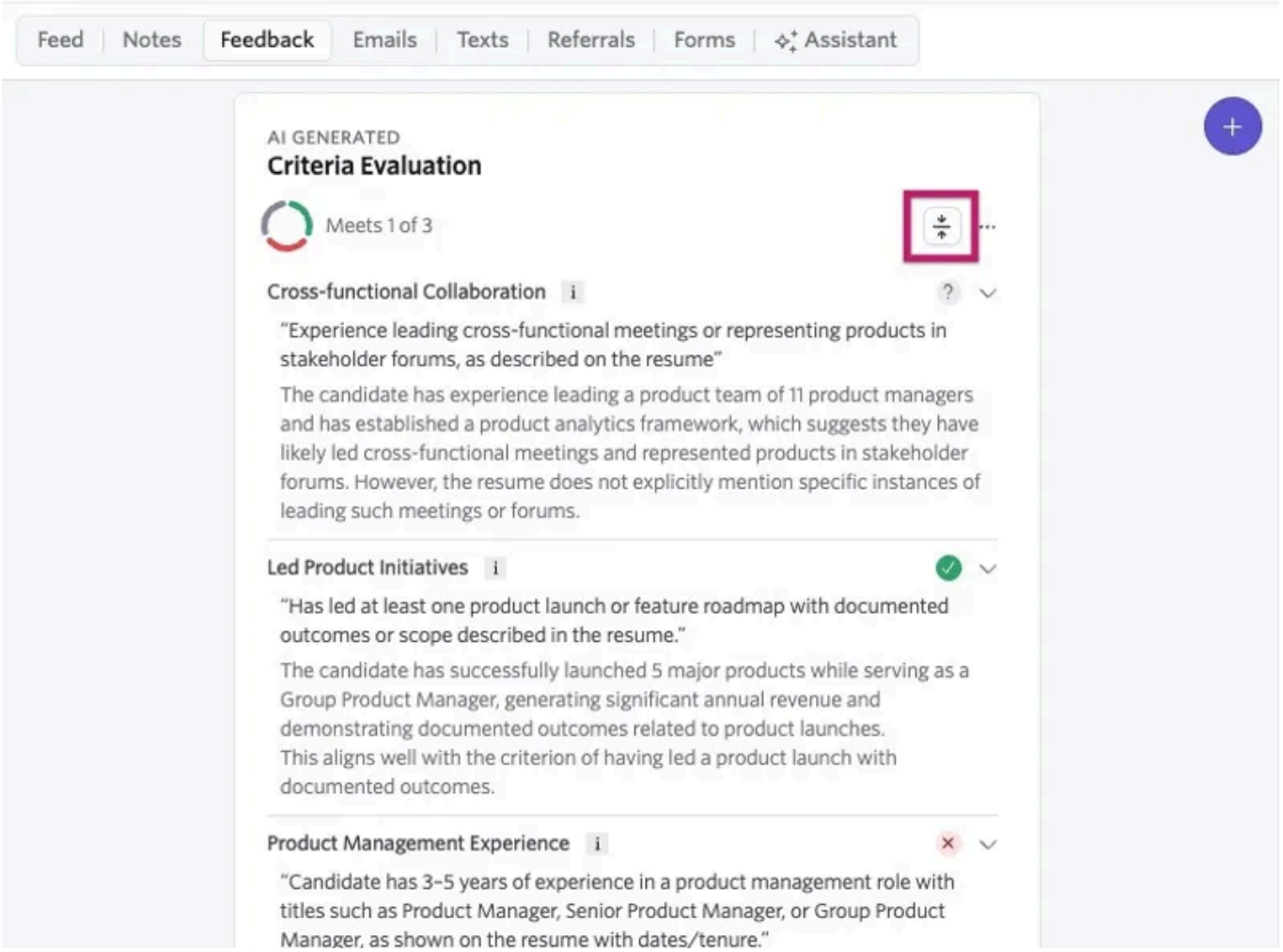
Ashby's AI assesses candidate resumes against each specific criterion and returns an evaluation status. The status may indicate the candidate meets, does not meet, or the AI was undecided on the criterion. Evaluation results can be seen on the candidate profile, as well as in the Application Review stage of the candidate pipeline for the role.

Viewing results on the candidate profile

View full evaluations in the **Feedback** tab of a candidate profile.

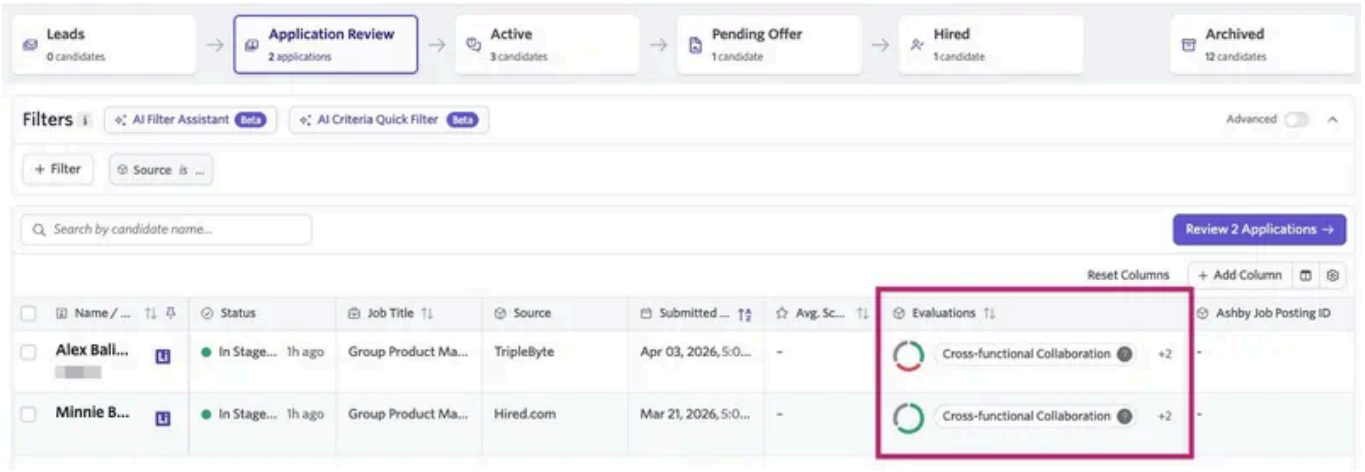


Click on any criterion to see the reasoning for a status of meets, does not meet, or undecided. Select the expand all icon to see reasoning for all criteria at once.



Evaluation results in Application Review

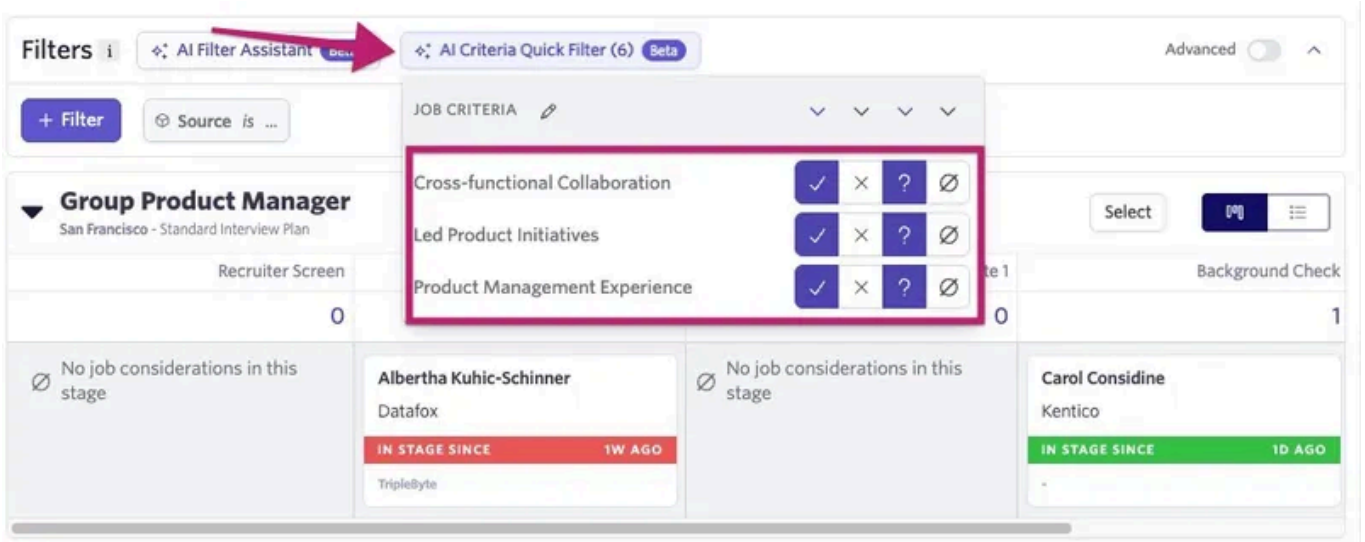
The Application Review stage of the pipeline shows a column for evaluation results that provides a summary about how the candidate scored. Hover over the results row to see more details.



Filtering by AI criteria

You can filter candidates by their evaluation status within your pipeline stages:

1. Select the `AI Criteria Quick Filter` option.
2. Select the criteria and statuses you want to view, such as any candidate who passed or the AI was undecided on a particular criteria.



3. Perform bulk actions on the filtered list as needed.

Automated processing legal notice

You can configure a message to display to candidates who may be impacted by automated AI review processes by following the steps below.

1. Navigate to **Admin > Data Management > Data Privacy and Compliance > Privacy & Legal** to configure a legal notice for candidates.
2. Locate the `Automated Processing Legal Notice` section and click on the `Legal Notice for All Jobs` that is displayed.
3. Configure the settings for when this message will display:
    - **Is Enabled?:** When off, candidates do not see the notice.
    - **Rule Scope:** Set criteria here when you want the message to show only based on certain criteria, such as for certain jobs only. Do not add a scope if you want it to apply to all jobs.
    - **Hide Scores:** If enabled, AI-informed scores are hidden from users if the candidate hasn't seen the notice.
    - **Legal Notice Text:** The specific text displayed to candidates. This can be edited.
4. Hit `Save` once you've adjusted the settings as needed.

Candidate experience

Candidates see a `Learn More` link under the `Submit Application` button. Clicking this allows them to view the notice, opt-out of automated processing, or check whether they have already opted-out. You can view what this page looks like by selecting `View` in the top right corner when setting up the message.

AI credits

AI-Assisted Application Review is priced per credit. One credit equals one evaluation of a single candidate (job consideration), supporting up to 50 criteria per evaluation.

Additional credits can be purchased in blocks. For more information, see our guide on **AI Credits**.

FAQ

If I edit my criteria, can I re-evaluate candidates?

Yes. If you edit a prompt or change criteria, select the Save and Re-Evaluate all active Job Considerations option to refresh evaluations.

Can I flag AI evaluations if they are incorrect?

Yes. Select the three dots in the top-right corner of an evaluation to flag it as incorrect or misleading.

Can I add evaluations as a column in Application Review?

Yes. Select `Add Column` on the application review page and choose **Job Consideration > Criteria > Evaluations**.

What permissions are needed to edit job criteria?

You must have Elevated Access - Admin or Elevated Access - Admin Private for the specific role, or be an Organization Admin.

Does the AI evaluate education and work history if there is no resume?

Yes, if the information is listed on the candidate's Ashby profile, the AI considers it.

17.1. Untitled

Get Cakes

Get a cake by its ID

GET https://api.cakes.com

Request

PATH PARAMETERS

Parameter name	`String`	`optional`

BODY PARAMETERS

id	`String`	`required`

ID of the cake to get

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");

var raw = JSON.stringify({
   "id": "string"
});

var requestOptions = {
   method: 'GET',
   headers: myHeaders,
   body: raw,
   redirect: 'follow'
};

fetch("https://api.cakes.com", requestOptions)
   .then(response => response.text())
   .then(result => console.log(result))
   .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```

18. this is a h1 title

@batch create

@get account st

19. Baguette parce que car cela ouais

test tables for CSS

column 1	col 2	col 3
papapapapapa	ddaawDASDSADASDAS	TEXT COLOR
HAHAHAHA	BEBEBEBEBE	CECECECECE
Added a new lin e	here	

Paste an image link or upload your image

Image URL

Add Image

19.1. Test doc

Nabucodonosor

asdasdas

1.0.7 OAS 2.0

Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key `special-key` to test the authorization filters.

Terms of service

Contact the developer

Apache 2.0

Find out more about Swagger

Schemes

HTTPS

Authorize

pet Everything about your Pets — Find out more

POST `/pet/{petId}/uploadImage` uploads an image

POST `/pet` Add a new pet to the store

PUT /pet Update an existing pet

GET /pet/findByStatus Finds Pets by status

GET ~~/pet/findByTags~~ Finds Pets by tags

Warning: Deprecated

GET /pet/{petId} Find pet by ID

POST /pet/{petId} Updates a pet in the store with form data

DELETE /pet/{petId} Deletes a pet

store Access to Petstore orders

GET /store/inventory Returns pet inventories by status

POST /store/order Place an order for a pet

GET /store/order/{orderId} Find purchase order by ID

DELETE /store/order/{orderId} Delete purchase order by ID

user Operations about user

Find out more about our store

POST /user/createWithList Creates list of users with given input array

GET /user/{username} Get user by user name

PUT /user/{username} Updated user

DELETE /user/{username} Delete user

GET /user/login Logs user into the system

GET /user/logout Logs out current logged in user session

POST /user/createWithArray Creates list of users with given input array

POST /user Create user

Models

- ApiResponse
- Category
- Pet
- Tag
- Order
- User

> Bucuresti

Lorem Ipsum: is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets

containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

20. Indenting

20.1. Another page

asdasdsadasdas

20.1.1. Advanced filtering

In the "faceting" section, we highlight how one can obtain available filters for a search query and apply filters to narrow down search results. For example, after searching for "shoes", a shopper may want to narrow down results by applying a filter on the color attribute and choosing a value such as red. Doing so will show only "red" shoes in the results. If at this point, the shopper chooses yet another color, e.g., blue, Klevu will now show both the "red" and the "blue" shoes. Thus, it will apply the OR operation between the values of the same attribute. If at this stage, the shopper chooses "cotton" as a material, the system will show both the red and blue shoes, but it will also ensure all products have "cotton" as a material as well. In other words, it applies the AND operation among the different types of selected attributes.

However, there comes a time when one may want to perform complex filtering operations, or in other words, advanced filtering operations. For example, you may want to show a group of shoes where the "color is red", and the "brand is NOT Adidas". You may also want to add another group of shoes to this result set. For example, where the "color is blue or black", "brand is Nike", and the "price range is between 100 and 400". As you can see, unlike the behaviour of "faceting", in our example here, we like to have one or more groups of products matching different groups of conditions for the final set of products to be returned.

It is with this intention to allow advanced filtering operations on the result set that we have introduced a new functionality called "advanced filtering" in our API.

Before the introduction of advanced filtering, if a user had requested filters to be returned or some to be applied, Klevu used to find out a set of relevant products matching the user query. It then included/excluded any products requested explicitly in this result set. It is after this stage that it calculates all possible filters to be returned and also applies the selected filters to narrow down the search results. Thus, if an explicitly added product did not match the applied filter, it would be excluded from the final result set.

With the addition of advanced filtering, Klevu now identifies a set of relevant products matching the query. It then applies advanced filtering to this. It is after this step that it includes/excludes any products requested explicitly, calculates filters based on this result set, and finally applies the selected filters to narrow down the search results. This way, if

there is any filtering to be applied prior to including/excluding products explicitly, advanced filtering should be used.

> Please note that the advanced filtering can also be used with Category Merchandising and Recommendation APIs.

The **groupCondition** object can be used for applying the advanced filtering conditions so your customers can fine-tune their results based on relevant attributes.

Apply Advanced Filtering

Example is as below:

POST https://{{hostname}}/cs/v2/search

Request

BODY PARAMETERS

groupOperator `String` `optional`

Defining the condition of filtering. Here are the options available.
ALL_OF:This is used to filter data using AND query i.e. the search results are matching with all the attributes.

ANY_OF:This is used to filter data using OR query i.e. the search results are matching with at least one attributes.

NONE_OF: This is used as negation AND query i.e. when search results are matching with none of the attribute values.

key `String` `optional`

The ID of the attribute to filter by, eg. color

valueOperator `String` `optional`

Defining an attribute's operation, Here are the options available.

INCLUDE : This is used to include specific attribute for the filter purpose.

EXCLUDE : This is used to exclude specific attribute for the filter purpose.

EXISTS: This is used to find if the specific indexed attribute value exists with product.

NOT_EXISTS : This is used to find if the specific indexed attribute value does not exist at all.

singleSelect `Boolean` `optional`

The behaviour when specifying multiple filters or values.

If it is set to "true", the products returned must have both the values.
If it is set to "false", the products returned must have at least one of the values.

excludeValuesInResult `Boolean` `optional`

For numeric filter like price, can be used to exclude data of the starting or ending values. For example if the range is defined as "1200 - 1232" in values, and excludeValuesInResult is set to true, Klevu will use range as “1201 - 1231”.

This can be used to achieve greater than or less than condition. For example,if you want to get the products greater than price 200, the values field will be “200 - *“ and the excludeValuesInResult will be set to true.

values `Array` `optional`

An array of values to filter by, eg. Red, Blue. When using range filters, specify the first value as the minimum value and the second as the max value. For example to retrieve records with prices between 60 to 80, use: "key": "klevu_price", "values": [60, 80].
To retrieve steps of prices, for example those with values between 0-50 or 150-200, use: "key": "klevu_price", "values": ["0-50", "150-200"].

To retrieve exact values, for example records with exact values 100 or 200, use: "key": "klevu_price", "values": ["100-100", "200-200"].
By default all attributes submitted to Klevu are indexed as STRING attributes, which means they cannot be used as range filters. The product sale price field is the only exception to this rule, which is filtered using the key klevu_price. If you have explicitly requested and Klevu has approved that certain attributes be indexed as numerical attributes, you can also retrieve those as range filters.

```
{
   "context":{
      "apiKeys":[
         "klevu-156925593843210765"
      ]
   },
   "recordQueries":[
      {
         "id":"productSearch",
         "typeOfRequest":"SEARCH",
         "settings":{
            "typeOfRecords":[
               "KLEVU_PRODUCT"
            ],
            "groupCondition":{
               "groupOperator":"ANY_OF"
```

```
{
    "meta": {
        "qTime": 9,
        "responseCode": 200
    },
    "queryResults": [
        {
            "id": "productSearch",
            "meta": {
                "qTime": 7,
                "noOfResults": 2,
                "totalResultsFound": 6,
                "typeOfSearch": "WILDCARD_AND",
                "offset": 0,
                "debuggingInformation": {},
                "notificationCode": 1,
                "searchedTerm": "tee",
                "apiKey": "klevu-156925593843210765",
                "isPersonalised": false
            },
            "records": [
                {
                    "discount": "",
                    "hideGroupPrices": "",
                    "type": "Tees",
                    "itemGroupId": "4384027344958",
                    "freeShipping": "",
                    "storeBaseCurrency": "GBP",
                    "price": "24.00",
                    "toPrice": "",
                    "imageUrl": "https://cdn.shopify.com/s/files/1/0116/9457/1582/pro
                    "currency": "GBP",
                    "inStock": "yes",
                    "id": "31366447038526",
                    "imageHover": "",
                    "sku": "WS05-XS-Yellow",
                    "brand": "KKE",
                    "basePrice": "24.0",
                    "startPrice": "",
                    "image": "https://cdn.shopify.com/s/files/1/0116/9457/1582/produc
                    "deliveryInfo": "",
                    "hideAddToCart": "",
                    "salePrice": "24.0",
```

```
"swatchesInfo": "",
"weight": "",
"klevu_category": "KLEVU_PRODUCT;Products;;Tees  @ku@kuCategory@k
"totalVariants": 0,
"groupPrices": "",
"url": "https://jsv2-shopify-demo.ksearchmisc.com/products/desire
"tags": "comfort, flattering fit, micro-sleeve, summer, v-neck",
```

20.2. New page

asdasdasdas

21. test page Ocient

The supports querying using the SQL syntax that follows ANSI SQL standards.

Ocient SQL Syntax

@virtual

This code block shows the general syntax to perform SQL querying in and the order in which commands should go.

For specific descriptions and syntax for the commands, see the respective SQL statement sections on this page.

Syntax

```
[ WITH ... ]
SELECT ...
    [ EXCEPT(...) ]
[ FROM ...
    [ JOIN ... ] ]
[ WHERE ... ]
[ GROUP BY ...
    [ HAVING ... ] ]
[ ORDER BY ... ]
[ LIMIT ... ]
[ OFFSET ... ]
[ INTERSECT ... ]
[ EXCEPT ... ]
[ UNION ... ] ]
[ USING ... ]
[ TRACE ... ]
[ TAG ... ]
```

Default Schema

Ocient identifies every table using a database and schema. For example, the fully qualified path to the `movies` table is `cinema.adventure.movies`, where `cinema` is the database and `adventure` is the schema.

When you do not fully qualify a table name, the Ocient System uses a default schema. When you first log into the system, the default schema is your fully qualified username. You can change the default schema using the **SET SCHEMA** command.

Querying SQL Statement Reference

Ocient supports the following SQL statements.

WITH

Assigns a name to a common table expression, allowing an auxiliary query to be used in the main query. This is helpful for breaking down complex queries into smaller parts.

Syntax

```
WITH
    <cte_name> [ ( <cte_column> [ ,... ] ) ]
    AS ( <sub_query> )
<select_query>
```

Parameters

Parameter	Description
`<cte_name>`	A name for the common table expression populated with the result set from the `sub_query` statement.
`<cte_column>`	Optional. This is a list of column names for common table expressions populated with the `sub_query` result set. If no column names are provided, the column names are the same as the table referenced in the `sub_query` statement.
`<sub_query>`	An auxiliary query that collects data for a common table expression. `sub_query` can use regular query commands, including ORDER BY, LIMIT, OFFSET, UNION, INTERSECT, EXCEPT, WHERE, GROUP BY, and HAVING. See the respective SQL statement reference sections for usage rules.
`<select_query>`	The main SELECT query. Note that the `FROM` statement in the main `SELECT` query must also include the `cte_name` if the query uses its values.

Example

In this example, the subquery calculates the average budget for all rows in the `movies` table. The main query uses that average to find all movies that spent more.

```
WITH avg_budget_table (average_budget) AS (
    SELECT AVG(budget)
    FROM movies
)
SELECT title,
    budget,
    revenue
FROM movies,
    avg_budget_table
WHERE budget > avg_budget_table.average_budget;
```

Output

title	budget	revenue
CGI Why	237000000	2787965087
Titania	200000000	1845034188
Merchandise Vehicle 5	200000000	1066969703
The Tentpole	220000000	1519557910
Pirates of Palm Springs	140000000	655011224
Spyman 16	200000000	1108561013
Frigid	150000000	1274219009
Fury 7	190000000	1506249360
Superhero 23	250000000	1084939099
Triassic World	150000000	1513528810
Iron Chef 3	200000000	1215439994
The Last Airman	150000000	318502923

SELECT

Initiates a query statement or a subquery clause within other statements.

You can query the data of tables where you have the SELECT privilege.

For information on using SELECT as a subquery for filtering or ordering results, see the **WHERE** and **HAVING** sections.

For information on using SELECT as a subquery for a common table expression, see the **WITH** section.

Syntax

```
SELECT [ ALL | DISTINCT ]
    [ * [ EXCEPT ( column_name [ , ... ] ) ]
    | <select_list_entry> [ , ... ] ]
    [ <from_clause> ]
```

Parameters

Parameter	Description
`ALL`	`SELECT` and `SELECT ALL` are both the same. Returns all valid data rows from the database that meet the criteria of your query.
`DISTINCT`	`SELECT DISTINCT` returns only unique rows that do not match other rows based on the criteria of your query.
`*`	Returns all columns from the specified tables in the result set for the query.
`EXCEPT`	When used with `*`, `EXCEPT` allows specific columns to be excluded from the query results.
`column_name`	The name of one or more columns from the specified table that you want to exclude (using `EXCEPT`) from your query results.

<select_list_entry>

The `select_list_entry` defines a column or expression to include in your query result set.

Syntax

```
<select_list_entry> ::=
    column_name | expression [ AS new_name ]
```

Parameters

Parameter	Description
`column_name`	The name of one or more columns from the specified table that you want to include in your query results.
`expression`	One or more expressions that you want to include in your query results. Expressions can be any combination of literal values, column names, arithmetic expressions, parentheses, and function calls.
`new_name`	An alias, treated as an identifier, for an alternative name for the column or the results of the expression.

<from_clause>

For information on the `<from_clause>`, see the **FROM** documentation.

Examples

Using SELECT *

This example uses `SELECT *` to return all the columns in the `movies` table.

```
SELECT * FROM movies LIMIT 5;
```

Output

movie_id	title	budget	popularity	release_date	revenue	runtime	movie_status	vote_average	vote_count
211672	Minnows	74000000	875.581305	2015-06-17	1156730962	91	Released	6.40	4571
24	Billy the Kilty	30000000	79.754966	2003-10-10	180949000	111	Released	7.70	4949
19995	CGI Why	237000000	150.437577	2009-12-10	2787965087	162	Released	7.20	11800
37724	Spyman 16	200000000	93.004993	2012-10-25	1108561013	143	Released	6.90	7604
24428	The Tentpole	220000000	144.448633	2012-04-25	1519557910	143	Released	7.40	11776

Using SELECT * EXCEPT

This example uses `SELECT * EXCEPT` to exclude certain columns from the result set.

```
SELECT *
EXCEPT (movie_id, runtime, vote_average, vote_count)
FROM movies
LIMIT 5;
```

Output

title	budget	release_date	revenue
Swords & Scabbards	94000000	2003-12-01	1118888979
Billy the Kilty	30000000	2003-10-10	180949000
Merchandise Vehicle 5	200000000	2010-06-16	1066969703
CGI Why	237000000	2009-12-10	2787965087
Space Odyssey 6000	10500000	1968-04-10	68700000

Using Lateral Column Aliases

Ocient SQL queries support lateral column aliases, meaning you can immediately reuse aliases for calculations in the same query as new inputs. Hence, you can simplify queries that normally require subqueries and common table expressions.

These examples use the `products` table with these columns:

- `product_id` — Product identifier as an integer
- `product_name` — Product name as a string
- `price` — Price as a floating point number

Create this table using the `CREATE TABLE` SQL statement.

```
CREATE TABLE products (
    product_id INT,
    product_name VARCHAR(100),
    price DECIMAL(10, 2)
    );
```

Insert four records into the `products` table.

```
INSERT INTO products (product_id, product_name, price) VALUES
    (1, 'Laptop', 1000.00),
    (2, 'Tablet', 500.00),
    (3, 'Smartphone', 800.00),
    (4, 'Monitor', 300.00);
```

Create a query to determine prices after discounts and taxes by using a common table expression subquery `DiscountedPrices`. Use the `WITH` keyword to create the subquery.

```
WITH DiscountedPrices AS (
    SELECT
        product_id,
        product_name,
        price,
        price * 0.90 AS discounted_price,
        price * 0.90 * 1.05 AS total_price_after_tax
    FROM
        products
)
SELECT
    product_id,
    product_name,
    price,
    discounted_price,
    total_price_after_tax
FROM
    DiscountedPrices;
```

Lateral aliases allow the same calculations to be packaged in a single query.

This simpler query is essentially the same as the longer common table expression example, but the logic is condensed because you can reference the `discounted_price` alias immediately to calculate the `total_price_after_tax` value in the same query.

```
SELECT
    product_id,
    product_name,
    price,
    price * 0.90 AS discounted_price,
    discounted_price * 1.05 AS total_price_after_tax
FROM
    products;
```

FROM

Specifies the table or view to use in a `SELECT` statement.

Syntax

```
<select_clause> FROM
    { table_name | ( <sub_query> ) } [ ,... ]
    [ <join_clause> [ ,... ] ]
```

Parameters

Parameter	Description
`<select_clause>`	A query statement using the `SELECT` SQL statement. For details, see the **SELECT** section.
`table_name`	The names of one or more tables or views that you want to query. If you specify multiple tables in the `FROM` list, separated by commas, the effect is the same as if you explicitly perform a **CROSS JOIN** on all of those sources.
`<sub_query>`	One or more subqueries using the `SELECT` SQL statement. A subquery generates a table from which the `select_clause` query references data. Each subquery must be enclosed in parentheses with an optional correlation name. For details, see the **SELECT** section.
`<join_clause>`	A `JOIN` clause used to retrieve data from two or more tables for your query. For details, see the **JOIN** section.

JOIN

Combines rows from multiple tables so they can be accessed by a query.

Syntax

```
<select_query>
    FROM <table_reference1> <join_operation> <table_reference2>
        ON table1_column <boolean_operator> table2_column
```

Parameter

Parameter	Description
`<select_query>`	A valid `SELECT` query. For details, see **SELECT**.
`<table_reference1>`	Either a table name or a full `SELECT` in parentheses `table_reference1` takes precedence for returning rows for `LEFT OUTER JOIN`, `SEMI JOIN`, and `ANTI JOIN`. For specific rules, see **Types of Join Operations**.
`<table_reference2>`	Either a table name or a full `SELECT` in parentheses function. `table_reference2` takes precedence for returning rows for `RIGHT OUTER JOIN`. For specific rules, see **Types of Join Operations**.
`table1_column`	A column from either `table_reference` The `JOIN` operation uses this column to match rows with `table2_column` to combine data for the result set.
`<boolean_operator>`	Join conditions can use any Boolean expression, including `=`, `!=`, `<`, `>`, `=>`, and `<=`
`table2_column`	A column from the `table_reference` not used for `table1_column` The `JOIN` operation uses this column to match rows based on the `table1_column` to combine data for the result set.

Types of Join Operations (`<join_operation>`)

Ocient supports the following types of `JOIN` operations.

Syntax

```
<join_operation> ::=
    { INNER JOIN
    | LEFT [ OUTER ] JOIN
    | RIGHT [ OUTER ] JOIN
    | FULL [ OUTER ] JOIN
    | CROSS JOIN
    | SEMI JOIN
    | ANTI JOIN }
```

JOIN Type Descriptions

JOIN Type	Description
`INNER JOIN`	Returns all rows that exist in both tables. This join type is the default type of join.
`LEFT OUTER JOIN`	Returns all rows from the first table regardless of whether there are matching rows in the second table. `LEFT JOIN` performs the same operation as `LEFT OUTER JOIN`.
`RIGHT OUTER JOIN`	Returns all rows from the second table regardless of whether there are matching rows in the first table. `RIGHT JOIN` performs the same operation as `RIGHT OUTER JOIN`.
`FULL OUTER JOIN`	Returns all matched and unmatched rows. `FULL JOIN` performs the same operation as `FULL OUTER JOIN`.
`CROSS JOIN`	Returns the Cartesian product of the joined tables. Each row in the first table is joined with all rows in the second table. The result set contains the number of rows in the first table multiplied by the number of rows in the second table.
`SEMI JOIN`	Returns the rows from the first table that have at least one match in the second table. No columns from the second table are available in the query.
`ANTI JOIN`	Returns the rows from the first table that have no matches in the second table. No columns from the second table are available in the query.

By default, `JOIN` statements that involve subqueries can operate laterally if necessary. This behavior allows subqueries to reference joined columns from the preceding items included in the `FROM` clause.

For example, both parts of this join operation use subqueries that reference table `x`. The `LATERAL` keyword is optional; the join operation acts the same regardless of whether it is included.

```
SELECT * FROM
    (
    SELECT
        1 AS cx1,
        2 AS cx2) AS x
INNER JOIN LATERAL (
    SELECT
        x.cx1 AS cy1,
        x.cx2 AS cy2) AS y
ON
    x.cx1 = y.cy1;
```

Lateral joins are primarily useful when a cross-referenced column is necessary for computing the rows to join. A common application is providing an argument value for a set-returning function.

Examples

These examples join two tables:

- `games` — A table of video game titles and their genre_ids. Note that some games have a NULL value assigned to their genre_id.

game_name	genre_id
Star Battle 6000	9
Dwarf Simulator	NULL
2002 Futbol	11
Space Attack!	9
Fantasy Fool	8
Blasto the Squirrel	5
Trucker Quest	7
Block, Stack, and Cry!	6
Basketball Day	11
Pizza Mutant	1
Italian Plumber	5
Zombies!	1
Space Ninjas	12
Moon Mayor	10
Spreadsheet Hero	NULL
Zoom Tycoon	12
Porch Poacher	6

- `genre` — A table of video game genres, which are identified by IDs.

genre_name	id
Racing	7
Puzzle	6
Adventure	2
Simulation	10
Shooter	9
Misc	4
Action	1
Sports	11
Role-Playing	8
Platform	5
Strategy	12
Fighting	3

INNER JOIN

This example uses an `INNER JOIN` operation to capture only the rows that exist in both tables. Games with NULL values for their `genre_id` are eliminated from the result set.

```
SELECT game_name,
    genre_name
FROM video_games.game
    INNER JOIN video_games.genre ON game.genre_id = genre.id;
```

Output

game_name	genre_name
Fantasy Fool	Role-Playing
Blasto the Squirrel	Platform
Italian Plumber	Platform
Porch Poacher	Puzzle
Space Ninjas	Strategy
Basketball Day	Sports
2002 Futbol	Sports
Moon Mayor	Simulation
Block, Stack, and Cry!	Puzzle
Zombies!	Action
Pizza Mutant	Action
Space Attack!	Shooter
Trucker Quest	Racing
Zoom Tycoon	Strategy

LEFT OUTER JOIN

This example uses a `LEFT OUTER JOIN` operation to capture all rows from the left table (game), even if they have no matching row in the right table (genre).

```
SELECT game_name,
    genre_name
FROM video_games.game
    LEFT OUTER JOIN video_games.genre ON game.genre_id = genre.id;
```

Output

game_name	genre_name
Fantasy Fool	Role-Playing
Space Ninjas	Strategy
Zoom Tycoon	Strategy
Zombies!	Action
Pizza Mutant	Action
Trucker Quest	Racing
Blasto the Squirrel	Platform
2002 Futbol	Sports
Basketball Day	Sports
Italian Plumber	Platform
Block, Stack, and Cry!	Puzzle
Star Battle 6000	Shooter
Dwarf Simulator	NULL
Moon Mayor	Simulation
Porch Poacher	Puzzle
Spreadsheet Hero	NULL
Space Attack!	Shooter

RIGHT OUTER JOIN

This example uses a `RIGHT OUTER JOIN` operation to capture all rows from the right table (genre), even if they have no matching row in the left table (game).

```
SELECT game_name,
    genre_name
FROM video_games.game
    RIGHT OUTER JOIN video_games.genre ON game.genre_id = genre.id;
```

Output

game_name	genre_name
NULL	Adventure
Trucker Quest	Racing
Pizza Mutant	Action
Moon Mayor	Simulation
Fantasy Fool	Role-Playing
Space Attack!	Shooter
NULL	Misc
Star Battle 6000	Shooter
Space Ninjas	Strategy
Porch Poacher	Puzzle
Block, Stack, and Cry!	Puzzle
Blasto the Squirrel	Platform
Italian Plumber	Platform
Zoom Tycoon	Strategy
NULL	Fighting
Zombies!	Action
2002 Futbol	Sports
Basketball Day	Sports

FULL OUTER JOIN

This example uses a `FULL OUTER JOIN` operation to capture all rows from both tables, even if they do not match.

```
SELECT game_name,
    genre_name
FROM video_games.game
    FULL OUTER JOIN video_games.genre ON game.genre_id = genre.id;
```

Output

game_name	genre_name
Basketball Day	Sports
2002 Futbol	Sports
Zombies!	Action
Blasto the Squirrel	Platform
Moon Mayor	Simulation
Trucker Quest	Racing
NULL	Misc
Star Battle 6000	Shooter
Space Attack!	Shooter
Spreadsheet Hero	NULL
NULL	Adventure
NULL	Fighting
Italian Plumber	Platform
Pizza Mutant	Action
Dwarf Simulator	NULL
Fantasy Fool	Role-Playing
Block, Stack, and Cry!	Puzzle
Space Ninjas	Strategy
Zoom Tycoon	Strategy
Porch Poacher	Puzzle

CROSS JOIN

This example uses a `CROSS JOIN` operation to capture every possible combination of rows from both tables, regardless of whether they match. Note that unlike other `JOIN` operations, `CROSS JOIN` does not require an `ON` statement.

```
SELECT game_name,
    genre_name
FROM video_games.genre
    CROSS JOIN video_games.game;
```

Output

> As the result set for this CROSS JOIN example is more than 200 rows, the results are abbreviated.

genre_name	genre_id
Spreadsheet Hero	Role-Playing
Spreadsheet Hero	Misc
Spreadsheet Hero	Sports
Spreadsheet Hero	Platform
Spreadsheet Hero	Strategy
Spreadsheet Hero	Shooter
Spreadsheet Hero	Fighting
Spreadsheet Hero	Action
Spreadsheet Hero	Simulation
Spreadsheet Hero	Racing
Spreadsheet Hero	Puzzle
Spreadsheet Hero	Adventure
Space Attack!	Role-Playing
Space Attack!	Misc
Space Attack!	Sports
Space Attack!	Platform
Space Attack!	Strategy
Space Attack!	Shooter
Space Attack!	Fighting
Space Attack!	Action
Space Attack!	Simulation
Space Attack!	Racing
Space Attack!	Puzzle
Space Attack!	Adventure
...	...

SEMI JOIN

This example uses a `SEMI JOIN` operation to capture genre names that have at least one match in the games table. Rows are only included once, even if there are multiple matches.

```
SELECT genre_name,
    id
FROM video_games.genre
    SEMI JOIN video_games.game ON genre.id = game.genre_id;
```

Output

genre_name	genre_id
Simulation	10
Shooter	9
Racing	7
Strategy	12
Puzzle	6
Platform	5
Action	1
Sports	11
Role-Playing	8

ANTI JOIN

This example uses an `ANTI JOIN` operation to capture any rows in the game table that do not match any rows in the genre table.

```
SELECT game_name
FROM video_games.game
    ANTI JOIN video_games.genre ON game.genre_id = genre.id;
```

Output

game_name
Spreadsheet Hero
Dwarf Simulator

WHERE

Filters rows based on a specified condition.

Syntax

```
<select_query>
    WHERE { column_name <filter_condition> filter_value |
    [ NOT ] EXISTS ( <sub_query> ) }
```

Parameters

Parameter	Description
`<select_query>`	A valid `SELECT` query.
`column_name`	A column used for grouping that is evaluated by the `filter_condition`.
`filter_value`	A value used to evaluate the specified `column_name`. For details, see the **<filter_condition>** section.
`<sub_query>`	A subquery with a filter condition that follows the `EXISTS` clause. For details, see the **EXISTS** section.

<filter_condition>

A logical combination of predicates used to evaluate the referenced `column_name` based on the `filter_value`.

Syntax

```
<filter_condition> ::=
  { =
    | ==
    | <>
    | !=
    | [ NOT ] EQUALS
    | <
    | <=
    | >
    | >=
    | [ NOT ] IN
    | [ NOT ] LIKE
    | [ NOT ] SIMILAR TO
    | BETWEEN
    | FOR SOME
    | FOR ALL
    | IS [ NOT ] NULL
    | IS [ NOT ] DISTINCT FROM
  }
```

Definitions

Operator	Description	Example
`=`, `==`, `EQUALS`	Equal to.	`Author = 'Alcott'`
`<>`, `!=`	Not equal to	`Dept <> 'Sales'`
`>`	Greater than	`Hire_Date > '2012-01-31'`
`<`	Less than	`Bonus < 50000.00`
`>=`	Greater than or equal	`Dependents >= 2`
`<=`	Less than or equal	`Rate <= 0.05`
`[ NOT ] LIKE`	Begins with a character pattern to match. The pattern is enclosed in parentheses and can include the wildcard characters `%`, representing zero, one or multiple characters, and `_` representing a single character. To have a literal `%` or `_` that does not act as a wildcard on the right-hand side of a `LIKE` or `NOT LIKE` expression, put a backslash before the character.	`Full_Name LIKE 'Will%'`
`[ NOT ] SIMILAR TO`	For usage information, see the **SIMILAR TO Operator** section.	`'abc' SIMILAR TO 'abc'`
`BETWEEN value1 AND value2`	Requires two values. The filter evaluates to TRUE if the specified column is within the range between the two values. `BETWEEN` bounds are inclusive. Values can be times, numbers, or strings.	`Sales BETWEEN 1000 AND 5000`
`FOR_SOME()`	Evaluates an array to determine if at least one value meets the filter criteria. In the example, the `FOR_SOME()` operator would return all rows of the column `col_int_array` where at least one value is greater than 10.	`FOR_SOME(col_int_array) > 10`

Operator	Description	Example
	For details, see **Array Filters**.	`FOR_SOME(col_int_array) > 10`

`FOR_ALL()`	Evaluates all values of an array to determine if all meet the filter criteria.
	In the example, the `FOR_ALL()` operator would return any rows of the column `col_int_array` where all values are greater than 10.
	For details, see **Array Filters**.
		`FOR_ALL(col_int_array) > 10`
		
		`FOR_ALL(col_int_array) = 1`

`[ NOT ] IN`	Equal to one of multiple possible values	`DeptCode IN (101, 103, 209)`
`IS [ NOT ] NULL`	Compare to `NULL` (missing data)	`Address IS NOT NULL`
`IS [ NOT ] DISTINCT FROM`	Compares the equality of two expressions for a Boolean result, including comparisons that involve `NULL` values. This operator ensures reliable comparisons in scenarios where `NULL` values need to be treated as significant data points. Normally, `NULL = NULL` evaluates to `false` because `NULL` represents an unknown value. However, the statement `a IS NOT DISTINCT FROM b` treats `NULL` as a comparable value and returns `true` when both values are `NULL` or when `a = b`. Inversely, `a IS DISTINCT FROM b` returns `true` if the values are different or if one value is `NULL` while the other is not `NULL`.	`Debt IS NOT DISTINCT FROM Receivables`

EXISTS

An `EXISTS` clause used in a `WHERE` statement evaluates if a subquery returns any rows.

For each row that the database computes in the outer query, if the subquery returns at least one row, the `EXISTS` clause evaluates to `true`. In this outcome, the outer query returns its row.

If the subquery returns zero rows, the `EXISTS` clause evaluates to `false`, and the outer query excludes those rows.

The `NOT EXISTS` clause performs the opposite Boolean logic. In this case, the outer query returns rows only when the subquery has no matches.

Examples

These examples use two tables, one for company departments and the other for employees who are assigned to those departments.

Create the `departments` table for department data.

```
CREATE TABLE departments (
    id INT,
    name VARCHAR(50)
);
```

Create the `employees` table for employee data.

```
CREATE TABLE employees (
    id INT,
    name VARCHAR(50),
    department_id INT
);
```

Insert department data for three departments, `HR`, `IT`, and `Marketing`, into the `departments` table.

```
INSERT INTO departments (id, name) VALUES
    (1, 'HR'),
    (2, 'IT'),
    (3, 'Marketing');
```

Insert employee data for four employees, `Alice`, `Bob`, `Charlie`, and `David`, into the `employees` table.

```
INSERT INTO employees (id, name, department_id) VALUES
    (1, 'Alice', 1),
    (2, 'Bob', 2),
    (3, 'Charlie', 2),
    (4, 'David', 4);
```

Find Employees Belonging to an Existing Department

This example uses an `EXISTS` clause to find any names from the `employees` table who are assigned to a department listed in the `departments` table.

```
SELECT name
FROM employees AS e
WHERE EXISTS (
    SELECT *
    FROM departments AS d
    WHERE e.department_id = d.id
);
```

The query returns all employees except `David`, who is assigned to a department that does not exist in the `departments` table.

Output

```
Bob
Charlie
Alice
```

Find Departments That Have Employees

This query returns departments that have at least one employee. The subquery uses `SELECT 1` to check whether any matching rows exist in the `employees` table. The output is the same if the query uses `SELECT *` instead.

```sql
SELECT name
FROM departments AS d
WHERE EXISTS (
    SELECT 1
    FROM employees AS e
    WHERE e.department_id = d.id
);
```

The query results exclude the `Marketing` department because no employees belong to it.

Output

```
HR
IT
```

Filter Departments Based on an Uncorrelated Table

In this example, the outer query filters the `departments` table where `id != 3` (excluding the `Marketing` department). The subquery checks if there is at least one employee with a department identifier less than `4`.

The query is uncorrelated because the subquery does not reference the `departments` table.

```
SELECT *
FROM departments d
WHERE d.id != 3
AND EXISTS (
    SELECT *
    FROM employees e
    WHERE e.department_id < 4
);
```

Output

```
HR
IT
```

Find Employees Without a Valid Department

This example uses a `NOT EXISTS` clause. The example returns only employees who are assigned to a department identifier `department_id` not listed in the `departments` table.

```
SELECT name
FROM employees AS e
WHERE NOT EXISTS (
    SELECT *
    FROM departments AS d
    WHERE e.department_id = d.id
);
```

Output: `David`

SIMILAR TO Operator

`SIMILAR TO` is a keyword that extends the `LIKE` operator, adding more features for match filtering including many metacharacters used in regular expressions.

`%` and `_` both act as wildcard operators, but other supported metacharacters match traditional regular expressions.

Syntax

```
WHERE string1 SIMILAR TO string2
```

This table describes the metacharacters supported by the `SIMILAR TO` keyword.

Metacharacter	Description
`%`	Repeated wildcard, matches any characters zero or more times. Equivalent usage to `LIKE`.
`_`	Wildcard, matches exactly one character. Equivalent usage to `LIKE`.
`|`	Alteration, meaning either of two alternatives (`a|b` represents `a` or `b`).
`*`	Repetition of zero or more times.
`+`	Repetition of one or more times.
`?`	Repetition of zero or one times.
`{m}`	Repetition of exactly `m` times.
`{m,n}`	Repetition between `m` and `n` times (inclusive).
`{m,}`	Repetition of at least `m` times.
`()`	Logical grouping.
`[]`	A character class, equivalent to character classes in regular expressions. Example: `[a-z]` is any lowercase English letter.
`^`	Beginning of the line anchor and the negation in character classes.
`$`	End of the line anchor.

`SIMILAR TO` also supports escaping these metacharacters with `\` and certain escape sequences supported by C family languages.

Metacharacter	Description
`\a`	The alert/bell character.
`\b`	The backspace character.
`\B`	A single `\` character. Equivalent to `\\`.
`\cX`	The character with the same five lower order bits as `X` with all zeros as higher order bits.
`\e`	The character with the numeric value 27, Esc.
`\f`	Form feed.
`\n`	Newline.
`\r`	Carriage return.
`\t`	Horizontal tab.
`\uwxyz`	`wxyz` represents four hexadecimal digits, including two characters: one with numeric value `wx`, and one with numeric value `yz`.
`\Uabcdefgh`	`abcdefgh` represents eight hexadecimal digits, including four characters: numeric value `ab`, numeric value `cd`, numeric value `ef`, and numeric value `gh`.
`\v`	Vertical tab.
`\xd....`	`d` is any number of hexadecimal digits. If the number of digits is odd, a 0 is added to the left of the sequence. Numeric characters with values represented by every group of two symbols from `d`.
`\0`	NULL character 0.
`\xy`	`xy` are octal digits that represent the character with the numeric value `xy`.
`\xyz`	`xyz` are octal digits that represent the character with the numeric value `xyz`. If `x` is greater than or equal to `4`, the database returns two characters: the numeric value `1` and the numeric value of the lower 8 bits of `xyz`.
`\d`	Matches any digit.
`\s`	Matches any whitespace.

Metacharacter	Description
`\w`	Matches any alphanumeric character or `_`.
`\D`	Matches any non-digit.
`\S`	Matches any non-whitespace.
`\W`	Matches any non-alphanumeric that is also not `_`.
`\A`	The beginning of a line. Equivalent to `^`.
`\m`, `\M`, or `\y`	The boundary of a word. Equivalent to `\b` in normal regular expressions.
`\Y`	Any position that is not the boundary of a word.
`\Z`	The end of a line. Equivalent to `$` in normal regular expressions.
`\*`	The star character.
`\+`	The plus character.
`\?`	The question mark character.

- The database ignores any `\` metacharacter that does not precede a metacharacter (including `\`) or an escape sequence in the metacharacter table.
- Invalid expressions (unpreceded quantifiers, unclosed parentheses, etc.) result in query errors.
- The database does not recognize the `.` character as a metacharacter, unlike the behavior in traditional regular expressions.
- The database applies escape metacharacters that apply to string literals. To match a string containing a single `'\'`, you might need to use `'\\\\'`.

Examples

```
'abc' SIMILAR TO 'abc'          true
'abc' SIMILAR TO 'a'            false
'abc' SIMILAR TO '%(b|d)%'      true
'abc' SIMILAR TO '(b|c)%'       false
'-abc-' SIMILAR TO '%\mabc\M%'  true
'xabcy' SIMILAR TO '%\mabc\M%'  false
```

Array Filters

A filter expression can use the functions `FOR_SOME()` and `FOR_ALL()` to apply a predicate against all values of an input array.

Array filter functions have the following rules:

- They can only evaluate array types.
- They can go on either the left or right side of a boolean comparison expression, but not both sides at the same time.
- They must directly use a boolean comparison operator.
- They cannot use the `SOME` and `ALL` SQL keywords.

Filter Behavior with Empty Arrays

Array filter functions have unique behavior when evaluating empty arrays.

- `FOR_ALL()` evaluates an empty array as `TRUE`.
- `FOR_SOME()` evaluates an empty array as `FALSE`.

If empty arrays must be evaluated for a different result, you can use the `ARRAY_LENGTH` function to specify a minimum array length. For example, this statement would evaluate an array as `TRUE` only if it is not empty and all values matched `%ocient%`:

```
ARRAY_LENGTH(array_col) > 0 AND FOR_ALL(array_col) LIKE '%ocient%
```

For details about array functions, see the **Array Functions and Operators** page.

Filter Behavior with NULL rows

Filtering with array functions can have different outcomes depending on whether the array row is NULL or if the values contained in the array are NULL.

If `FOR_SOME()` or `FOR_ALL()` evaluate a NULL row (the row itself is NULL, not that the array contains NULL values), then the result is always `FALSE`.

To check an array for the presence of NULL values, you must use the `IS NULL` operator. For example:

```
FOR_SOME(array_col) IS NULL
```

Array comparison operators, such as `@>`, `<@`, and `&&` do not adhere to Boolean logic for NULL values. For details, see the **Array Functions and Operators** page.

GROUP BY

Groups rows with the same values into summary rows, based on a specified aggregate function.

Syntax

```
SELECT <select_list_clause> FROM table_name
    GROUP BY { column_name | expression | integer } [ , ... ]
    [ HAVING <filter_condition> ]
```

Parameters

Parameter	Description
`<select_list_clause>`	A list of one or more columns for the `SELECT` query. To use `GROUP BY` to summarize the query result set, at least one of the columns in `column_list` must include an aggregate function, such as `SUM()`, `COUNT()`, or `MAX()`. For a list of supported functions, see the **Aggregate Functions** page.
`table_name`	The table to use for the query.
`column_name`	The name of one or more columns to use to group the result set. If you specify multiple columns, the result set is grouped by each unique combination of values from the columns.
`expression`	Any combination of literal values, column names, arithmetic expressions, parentheses, and function calls.
`integer`	An integer representing the position of the columns referenced in the `column_list`. The first position starts at `1`.
`<filter_condition>`	A `HAVING` clause that filters the aggregated groups based on a specified condition. For details, see the **<filter_condition>** section.

Example

This example uses a movie database to calculate the total amount spent on movie production per year.

```
SELECT YEAR(release_date),
    SUM(budget)
FROM movies
GROUP BY 1
ORDER BY 1 ASC;
```

Output

year(release_date)	sum(budget)
1968	10500000
1979	31500000
1981	18000000
1982	28000000
1992	14000000
1997	200000000
2003	264000000
2009	237000000
2010	350000000
2012	670000000
2013	350000000
2015	414000000

HAVING

Filters aggregated rows based on a specified condition.

`HAVING` operates in a `GROUP BY` statement by setting a filter for the rows to be aggregated and grouped.

For information on using `GROUP BY` in a query statement, see **GROUP BY**.

Syntax

```
GROUP BY column_name HAVING <filter_condition>
```

Parameters

Parameter	Description
`column_name`	A column used for grouping that is evaluated by the `filter_condition`.
`<filter_condition>`	A logical combination of boolean predicates. For information on supported logical predicates, see **<filter_condition>**.

Example

This example calculates the total amount spent on movie production per year. The `HAVING` clause filters out any rows that do not have a sum of at least $100 million.

```
SELECT YEAR(release_date),
    SUM(budget)
FROM movies
GROUP BY 1
HAVING SUM(budget) > 100000000
ORDER BY 1 ASC;
```

Output

year(release_date)	sum(budget)
1997	200000000
2003	264000000
2009	237000000
2010	350000000
2012	670000000
2013	350000000
2015	414000000

ORDER BY

Sorts the result set in ascending or descending order based on one or more specified columns.

If you specify multiple columns, they are sorted hierarchically from left to right.

Syntax

```
ORDER BY { column_position | column_name } [ ASC | DESC ]
    [ NULLS FIRST | NULLS LAST ] [ , ... ]
```

Parameters

Parameter	Data Type	Description
`column_position`	Integer	The position of a column to be used for the `ORDER BY` sorting. Column positions start at `1`.
`column_name`	String	The name of a column to be used for the `ORDER BY` sorting.
`ASC | DESC`	String	Optional. Specifies whether to sort the column in ascending (`ASC`) or descending (`DESC`) order. If unspecified, defaults to `ASC`.
`NULLS FIRST | NULLS LAST`	String	Optional. `NULLS FIRST` means null values go at the start of the result set. `NULLS LAST` means null values go at the end of the result set. If unspecified, defaults to `NULLS FIRST`.

Example

In this example, the `ORDER BY` statement orders the movies from newest to oldest.

```
SELECT release_date, title
FROM movies
ORDER BY release_date DESC;
```

Output

release_date	title
2015-06-17	Minnows
2015-06-09	Triassic World
2015-04-01	Fury 7
2013-11-27	Frigid
2013-04-18	Iron Chef 3
2012-10-25	Spyman 16
2012-07-16	Superhero 23
2012-04-25	The Tentpole
2010-06-30	The Last Airman
2010-06-16	Merchandise Vehicle 5
2009-12-10	CGI Why
2003-12-01	Swords & Scabbards
2003-10-10	Billy the Killy
2003-07-09	Pirates of Palm Springs
1997-11-18	Titania
1992-08-07	Bang Bang Western
1982-06-25	Blade Walker
1981-06-12	Raiders of the Jungle Temple
1979-08-15	Apocalypse None
1968-04-10	Space Odyssey 6000

LIMIT

Limits the number of returned rows from a query to a specified amount.

The returned rows are nondeterministic unless you use an **ORDER BY** clause.

Syntax

```
<select_query> LIMIT limit_number
```

Parameters

Parameter	Description
`<select_query>`	A valid `SELECT` query.
`limit_number`	The number of rows to return from the query, specified as a positive integer or any constant scalar expression that evaluates to a positive integer.

Examples

Limit Returned Rows Using a Number

This example uses `LIMIT` to restrict the number of returned movie titles to only three.

```
SELECT title FROM movies LIMIT 3;
```

Output

```
Minnows
CGI Why
Billy the Killy
```

Limit Returned Rows Using an Expression

The `LIMIT` SQL statement can also accept expressions. This query uses the `1+2` expression with the `sys.dummy` table to create a table of three incrementing integers. For details, see **Generate Tables Using sys.dummy**.

```
SELECT c1 FROM sys.dummy10 LIMIT 1+2;
```

Output

```
1
2
3
```

OFFSET

Skips a specified number of rows from the result set.

The returned rows are nondeterministic unless you use an **ORDER BY** clause.

Syntax

```
<select_query> OFFSET offset_number
```

Parameters

Parameter	Description
`<select_query>`	A valid SELECT query.
`offset_number`	The number of rows to skip when returning results from the query, specified as a positive integer or any constant scalar expression that evaluates to a positive integer.

Examples

Skip Rows From the Result Set Using a Number

In this example, the `ORDER BY` statement orders the results chronologically. As a result, the `OFFSET` clause removes the oldest six movies from the result set.

```
SELECT release_date,
    title
FROM movies
ORDER BY release_date OFFSET 6;
```

Output

release_date	title
2003-07-09	Pirates of Palm Springs
2003-10-10	Billy the Killy
2003-12-01	Swords & Scabbards
2009-12-10	CGI Why
2010-06-16	Merchandise Vehicle 5
2010-06-30	The Last Airman
2012-04-25	The Tentpole
2012-07-16	Superhero 23
2012-10-25	Spyman 16
2013-04-18	Iron Chef 3
2013-11-27	Frigid
2015-04-01	Fury 7
2015-06-09	Triassic World
2015-06-17	Minnows

Skip Rows From the Result Set Using an Expression

The `OFFSET` SQL statement can also accept expressions. This query uses the expression `3+4` with the `sys.dummy` table to create a column of incrementing integers by skipping the first seven out of 10 rows. For details, see **Generate Tables Using sys.dummy**.

```
SELECT c1 FROM sys.dummy10 OFFSET 3+4;
```

Output

```
8
9
10
```

INTERSECT

Returns any rows that match between two separate `SELECT` queries. To use `INTERSECT`, the two queries must be compatible, meaning they must return the same number of columns and have similar data types.

By default, the SQL statement eliminates duplicate rows unless the query includes the optional `ALL` keyword. Using the `DISTINCT` keyword is the same as this default behavior.

Syntax

```
<select_query1> INTERSECT [ ALL | DISTINCT ] <select_query2>
```

Parameters

Parameter	Description
`<select_query1>`	A valid SELECT query.
`<select_query2>`	A valid SELECT query.

Example

This example has two `SELECT` queries with an `INTERSECT` statement. The full query retrieves movies that had a budget of at least $200 million, but also earned more than $1 billion in revenues.

```sql
SELECT *
FROM movies
WHERE revenue > 1000000000
INTERSECT
SELECT *
FROM movies
WHERE budget > 20000000;
```

Output

movie_id	title	budget	popularity	release_date	revenue	runtime	movie_status	vote_average	vote_count
597	Titania	200000000	100.025899	1997-11-18	1845034188	194	Released	7.5	7562
19995	CGI Why	237000000	150.437577	2009-12-10	2787965087	162	Released	7.2	11800
49026	Superhero 23	250000000	112.31295	2012-07-16	1084939099	165	Released	7.6	9106
10193	Merchandise Vehicle 5	200000000	59.995418	2010-06-16	1066969703	103	Released	7.6	4597
211672	Minnows	74000000	875.581305	2015-06-17	1156730962	91	Released	6.4	4571
122	Swords & Scabbards	94000000	123.630332	2003-12-01	1118888979	201	Released	8.1	8064
37724	Spyman 16	200000000	93.004993	2012-10-25	1108561013	143	Released	6.9	7604
109445	Frigid	150000000	165.125366	2013-11-27	1274219009	102	Released	7.3	5295
168259	Furious 7	190000000	102.322217	2015-04-01	1506249360	137	Released	7.3	4176
68721	Iron Chef 3	200000000	77.68208	2013-04-18	1215439994	130	Released	6.8	8806
135397	Triassic World	150000000	418.708552	2015-06-09	1513528810	124	Released	6.5	8662
24428	The Tentpole	220000000	144.448633	2012-04-25	1519557910	143	Released	7.4	11776

EXCEPT

The `EXCEPT` keyword returns the result set of a first `SELECT` query minus any matching rows from a second `SELECT` query.

`EXCEPT` requires the two queries to be compatible, meaning they must return the same number of columns and have similar data types. By default, the result set eliminates duplicate rows unless the query includes the optional `ALL` keyword. Using the `DISTINCT` keyword is the same as this default behavior.

The `EXCEPT` keyword can also be used to exclude specific columns from a `SELECT *` query. For information on that alternate usage, see the **SELECT** syntax and example.

Syntax

```
<select_query1> EXCEPT [ ALL | DISTINCT ] <select_query2>
```

Parameters

Parameter	Description
`<select_query1>`	A valid SELECT query that is compared to `select_query2`. The results of an EXCEPT statement are the rows in `select_query1` that do not match any row in `select_query2`.
`<select_query2>`	A valid SELECT query that is compared to `select_query1`.

Example

In this example, the first `SELECT` statement queries all rows in the `movies` table. The `EXCEPT` clause and the second `SELECT` statement eliminate from the results any rows with movies that had a budget greater than `20000000`.

```
SELECT *
FROM movies
EXCEPT
SELECT *
FROM movies
WHERE budget > 20000000;
```

Output

title	budget	popularity	release_date	revenue	runtime	movie_status	vote_average	vote_count
Bang Bang Western	140000000	37.380435	1992-08-07	159157447	131	Released	7.7	1113
Space Odyssey 6000	105000000	86.201184	1968-04-10	68700000	149	Released	7.9	2998
Raiders of the Jungle Temple	180000000	68.159596	1981-06-12	389925971	115	Released	7.7	3854

UNION

Returns the combined result set of two or more `SELECT` queries.

By default, `UNION` eliminates duplicate rows from the result set unless you specify the `ALL` keyword. Using the `DISTINCT` keyword is the same as this default behavior.

Syntax

```
<select_query1> UNION [ ALL | DISTINCT ] <select_query2> [ ... ]
```

Parameters

Parameter	Data Type	Description
`select_query1`	String	A valid `SELECT` query that is combined with `select_query2`.
`select_query2`	String	A valid `SELECT` query that is combined with `select_query1`.

Example

This example uses `UNION` to merge two separate queries for identical columns into the same result set.

```
SELECT title,
    popularity,
    vote_average
FROM movie
WHERE popularity > 800
UNION
SELECT title,
    popularity,
    vote_average
FROM movies.movie
WHERE vote_average > 7.5;
```

Output

title	popularity	vote_average
Raiders of the Jungle Temple	68.159596	7.7
Minnows	875.581305	6.4
Apocalypse None	49.973462	8
Superhero 23	112.31295	7.6
Billy the Killy	79.754966	7.7
Bang Bang Western	37.380435	7.7
Blade Walker	94.056131	7.9
Swords & Scabbards	123.630332	8.1
Merchandise Vehicle 5	59.995418	7.6
Space Odyssey 6000	86.201184	7.9

USING

Overrides various system configurations for processing a specified query.

The `USING` keyword is required for only the first query override, not for subsequent ones.

For descriptions of the supported query configurations, see the parameter table below.

Syntax

```
<select_query>
USING
    [ SCHEDULING_PRIORITY = priority_value ]
    [ MAX_ROWS_RETURNED = max_rows ]
    [ MAX_ELAPSED_TIME = max_elapsed_time ]
    [ MAX_TEMP_DISK_USAGE = max_temp_disk_usage ]
    [ CACHE_MAX_TIME = cache_max_time ]
    [ CACHE_MAX_BYTES = cache_max_bytes ]
    [ SERVICE CLASS service_class ]
```

Parameters

Parameter	Description
`<select_query>`	A valid `SELECT` query.
`priority_value`	An optional clause that specifies the priority at which the query should run. The priority value is a floating point number, which indicates the priority compared to the other queries. The priority value does not specify any percentage of resources that the query takes or any specific degree of difference between queries. If unspecified, defaults to `SCHEDULING_PRIORITY = 1.0`.
`max_rows`	An optional clause that limits the number of rows returned by a query. If a query exceeds this number of rows, the system kills it. For example, `max_rows_returned = 10` kills any queries returning more than ten rows.
`max_elapsed_time`	An optional clause that limits how long a query can run. If a query exceeds this time limit in seconds, the system kills it. For example, `max_elapsed_time = 10` kills any queries taking longer than 10 seconds.
`max_temp_disk_usage`	An optional clause that limits the percentage of temporary disk used by a query. If a query exceeds this temporary disk limit, the system kills it. For example, `max_temp_disk_usage = 10` kills any queries taking more than 10% of temporary disk space.
`cache_max_time`	An optional clause that affects whether a specified query uses results from the cache rather than executing the query. If there is a cached result with the same query text, executed less than `cache_max_time` seconds ago, the database uses that cached result. The default value is 0, and thus by default, the database does not return any cached results. The cache considers all SQL Nodes, and if a potential cached result is only available on a different SQL Node, the database redirects the query to that node.

Parameter	Description
	If you specify the force attribute on the connection, which disables load balancing and redirection, the database considers only cached results on the current node.
`cache_max_bytes`	An optional clause that controls whether the database stores the results of any specified query in the cache. The system stores the results of all queries executed using a service class specifying this attribute in the cache if the result size is smaller than this value. You can determine the size of a result set in bytes by querying the `bytes_returned` field of the `completed_queries` virtual table. The default value is 0, and thus by default, the database does not cache any results. The database caches results in memory on the SQL Nodes. These results are not cached if there is insufficient memory available.
`service_class`	An optional clause at the end of a query that sets a specific service class to run that query. For details, see **CREATE SERVICE CLASS**.

Example

```
SELECT * FROM sys.dummy10
USING
    SCHEDULING_PRIORITY = 1.0
    MAX_ROWS_RETURNED = 10
    MAX_ELAPSED_TIME = 10
    MAX_TEMP_DISK_USAGE = 10;
```

Output

```
c1
-----------
1
2
3
4
5
6
7
8
9
10
```

TRACE

An optional clause that executes the query, but discards the original result set. Instead, `TRACE` returns a result set of tracing data that describe the execution of the query.

Append the `TRACE` SQL statement to the end of a query. The statement can include optional parameters to control its frequency and level of detail.

To understand the results of this statement, see **TRACE Results**.

> Trace queries have the same effect to the workload as a query run without the clause.

> Contact Ocient Support for guidance in using the TRACE SQL statement.

Syntax

```
<select_query> TRACE
    [ FREQUENCY frequency_int ]
    [ RESOLUTION resolution_int ]
```

Parameters

Parameter	Description
`<select_query>`	A valid `SELECT` query.
`frequency_int`	Controls how often the database samples trace events during query execution in milliseconds. If unspecified, defaults to 500 milliseconds.
`resolution_int`	Determines the level of detail captured for a trace or event. If the resolution of a trace is greater than the resolution of an event, the database records the event. If unspecified, defaults to 100 milliseconds.

TRACE Results

Each row of the trace output provides some information about what an operator instance has done in the time between the previous trace sample and the most current sample.

Column	Description
`plan_parent_id`	The UUID of the parent of the operator instance in the query plan. If an operator has multiple parents, this field specifies the UUID of one of the parents.
`plan_op_id`	The UUID of the operator in the query plan.
`operator_type`	The type of operator that the database uses to sample trace events.
`node_id`	The UUID of the node where this operator executes.
`code_id`	The identifier of the VM core where this operator instance executes. Core identifiers are unique for each node.
`op_id`	The identifier of the operator instance. Identifiers are unique for each node.
`time`	The time in milliseconds when the database collects the information for the trace event relative to the start of the query.
`group`	Group related trace events together.
`sample`	The sample of trace event data. For possible values, see the following table.
`value`	The value of trace event data.

Sample and Value Columns

The `sample` and `value` columns describe what occurred in a given trace period.

Sample	Description
`ROWS_IN`	The number of rows an operator instance has read from its children operators.
`ROWS_OUT`	The number of rows the operator instance returns.
`BLOOM_FILTERED_ROWS`	The number of rows that this operator instance discards by using Bloom filters.
`SCHEDULE_CYCLE`	The number of cycles in the schedule for this operator instance.
`OOM_CYCLE`	The number of out-of-memory cycles in the schedule for this operator.
`INITIALIZE`	The time when the database initializes the operator.
`FINALIZE`	The time when the database finalizes the operator.

Example

```
SELECT * FROM sys.dummy10 TRACE;
```

Output

plan_parent_id	plan_op_id	operator_type	node_id	core_id	op_id	time	group	sample	value
77041d2b-f7ca-49d5-a3f5-4b75b0782ed1	498690cf-4a9b-4c1f-affa-417f31fddc2d	RENAME_OPERATOR	08ca7b05-1e4d-455f-9f46-2174ed048d33	4	8441	0	xg::db::vm::operators::operatorTraceEvents	ROWS_IN	1
77041d2b-f7ca-49d5-a3f5-4b75b0782ed1	498690cf-4a9b-4c1f-affa-417f31fddc2d	RENAME_OPERATOR	08ca7b05-1e4d-455f-9f46-2174ed048d33	4	8441	0	xg::db::vm::operators::operatorTraceEvents	ROWS_OUT	1
77041d2b-f7ca-49d5-a3f5-4b75b0782ed1	498690cf-4a9b-4c1f-affa-417f31fddc2d	RENAME_OPERATOR	08ca7b05-1e4d-455f-9f46-2174ed048d33	4	8441	0	xg::db::vm::operators::operatorTraceEvents	BLOOM_FILTERED_ROWS	0

TAG

An optional clause that adds one or more tags to the SQL query. You can also add tags to a common table expression in the `WITH` clause. After you define tags, you can find them in the `sys.queries` and `sys.completed_queries` system catalog tables.

Syntax

```
<select_query> <tag> [ ... ]

<tag> ::= TAG <tag_identifier>
```

Parameters

Parameter	Description
`<select_query>`	A valid `SELECT` query.
`tag_identifier`	Identifier of the tag for the query. Enclose the identifier in double quotes if it contains any special characters, such as spaces.

Examples

SQL Query with One Tag

Select the count of a table with one row and tag the query with the name `count1`.

```
SELECT COUNT(*) FROM sys.dummy1 TAG count1;
```

SQL Query with Multiple Tags

Select the device model and tag this query with two names `device_model` and `phones`.

```
SELECT device_model
FROM system.adtech_flat
LIMIT 1
TAG device_model
TAG phones;
```

Common Table Expression with One Tag

Define a common table expression with the `average_budget` tag. You can also add another tag for the whole query `overall_budget`.

```
WITH avg_budget_table (average_budget) AS (
    SELECT AVG(budget)
    FROM movies
    TAG average_budget
)
SELECT title,
    budget,
    revenue
FROM movies,
    avg_budget_table
WHERE budget > avg_budget_table.average_budget
TAG overall_budget;
```

String Literals and Escape Sequences

All string literals in SQL statements must be enclosed in single quotes. To use a single quote within a string, you can use another single quote as an escape, `''`.

For other escape sequences, include an `e` character before the string literal, i.e., before the opening single quote. This directs the system to recognize escape sequences in the string literal. This means:

- All single `\` characters in the string now escape themselves.
- Any subsequent character after the `\` is also escaped if it matches an escape sequence (see the table).

If you use escape sequences, be prepared that you might need to alter strings that include `\`, such as directory paths.

Supported Escape Sequences

Escape Sequence	Description
`''`	Single quotation mark
`\"`	Double quotation mark
`\n`	Newline character
`\r`	Carriage return character
`\f`	Form feed character (i.e. page break)
`\b`	Backspace
`\\`	Backslash
`\t`	Tab

Examples

These examples show how the system interprets strings with and without escape sequences.

String Without an Escape Sequence

This example selects the simple string literal `'\my\directory\path'` that does not use escape sequences.

```
SELECT '\my\directory\path';
```

Output: `\my\directory\path`

String With an Escape Sequence

This example selects the simple string literal `'\my\directory\path'` and uses an escape sequence. The result omits the backslashes.

```
SELECT e'\my\directory\path';
```

Output: `mydirectorypath`

String With an Escape Sequence to Retain the Backslashes

This example uses the `'\\my\\directory\\path'` string with an escape sequence to include backslashes in the result.

```
SELECT e'\\my\\directory\\path';
```

Output: `\my\directory\path`

Escape Sequences with Regular Expressions

> Be careful using escape sequences with strings that also use regular expressions. Escape sequences in SQL syntax override the escape sequences of regular expressions.

In this example, the **REGEXP_SUBSTR** function uses the regular expression `\w+` to match any word characters.

```
SELECT REGEXP_SUBSTR(
    'abcdefghijklmnopqrstuvwxyz',
    '\w+'
    );
```

Output: `abcdefghijklmnopqrstuvwxyz`

The function behaves differently if the regular expression is an escape sequence because it overrides the `\` character. The regular expression searches only for the character `w`.

```
SELECT REGEXP_SUBSTR(
    'abcdefghijklmnopqrstuvwxyz',
    e'\w+'
    );
```

Output: `w`

Related Links

SQL Reference

Data Control Language (DCL) Statement Reference

Data Definition Language (DDL) Statement Reference

22. Reminder

22.1. Toxic Language

The toxic language guardrail detects harmful content including hate speech, threats, insults, and other communication that could damage your community or brand. Unlike simple keyword filters, this guardrail uses AI to understand context, tone, and intent.

knmknknkjn

Paste an image link or upload your image

Image URL

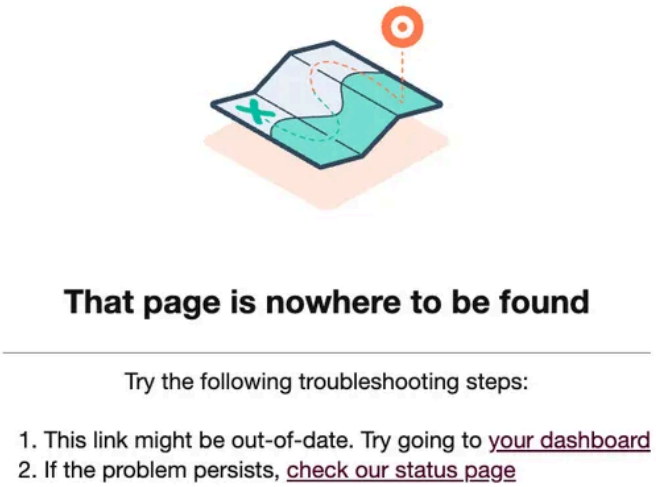
Add Image

When to Use This Guardrail

Paste an image link or upload your image

Image URL

Add Image



The key advantage over simple keyword filtering is that this guardrail understands nuance. Someone can be hostile without using profanity, and they can use strong language without being hostile. The guardrail analyzes intent and tone, not just word choice.

Understanding Sensitivity Levels

The sensitivity setting controls how strict the guardrail is. Think of it as adjusting the threshold for what constitutes a violation. This is one of the most important configuration choices you'll make because it fundamentally changes what content passes or fails.

Low sensitivity flags only severe violations like explicit threats and extreme hate speech. When you set sensitivity to low, you're saying that you expect strong opinions and robust disagreement, and you only want to block content that crosses a clear line into threatening or hateful territory. For example, "I strongly disagree with this approach and think it's misguided" would pass at low sensitivity, as would "This is a terrible idea." Only content like "I will find you and hurt you" or explicit hate speech would fail.

Low sensitivity works well for public forums where debate is expected, professional feedback environments where directness is valued, and technical communities where people discuss contentious topics. The tradeoff is that some content that makes people uncomfortable might still pass through.

Medium sensitivity is the default setting and represents a balanced approach. At this level, the guardrail flags clear violations including insults and hostile language while still allowing professional disagreement and constructive criticism. A message like "I disagree with your reasoning" would pass, but "You're an idiot" or "People like you are the problem" would fail.

Medium sensitivity works well for most applications including customer service systems, business communications, collaborative tools, and social platforms. It strikes a balance between allowing meaningful discourse and maintaining a respectful environment.

High sensitivity creates the strictest environment by flagging any potentially toxic content including mild rudeness and dismissive language. At this level, even content like "Whatever, dude" or "That's pretty dumb" would fail. Only respectful, neutral content passes at high sensitivity.

High sensitivity is appropriate for children's applications where you need maximum protection, educational platforms where you want to model respectful communication, safe spaces and support communities where people need to feel secure, and compliance-critical contexts where any potential issue needs to be caught.

Configuration Options

The toxic language guardrail accepts several configuration options that let you tune its behavior for your specific use case.

```
// Available options:
// sensitivity: "low", "medium", or "high" (default: "medium")
// model: model identifier (default: Claude 3.5 Haiku)
// temperature: 0-1, lower values = more consistent (default: 0.1)
// maxTokens: response length limit (default: 200)

await abv.guardrails.toxicLanguage.validate(text, {
  sensitivity: "medium",
  model: "model-name",
  temperature: 0.1,
  maxTokens: 200,
});
```

```
# Available options:
# sensitivity: "low", "medium", or "high" (default: "medium")
# model: model identifier (default: Claude 3.5 Haiku)
# temperature: 0-1, lower values = more consistent (default: 0.1)
# maxTokens: response length limit (default: 200)

abv.guardrails.toxic_language.validate(text, {
    "sensitivity": "medium",
    "model": "model-name",
    "temperature": 0.1,
    "maxTokens": 200
})
```

The sensitivity option is the most important and you'll use it frequently. The model, temperature, and maxTokens options are advanced settings that you typically won't need to change. The default model is optimized for guardrail tasks and provides the best balance of speed, accuracy, and cost. The default temperature of 0.1 ensures consistent results. The default maxTokens of 200 is sufficient for the explanation field.

Real-World Examples

Let's look at concrete examples of how different sensitivity levels handle various types of content. Understanding these patterns will help you choose the right sensitivity for your application.

Consider a message like "I disagree with your approach to this problem." This is professional disagreement and passes at all sensitivity levels. The language is neutral and respectful despite expressing disagreement.

Now consider "This is a terrible idea and shows poor judgment." This passes at low and medium sensitivity because while it's critical, it focuses on the idea rather than attacking the person. However, it might fail at high sensitivity because "terrible" and "poor judgment" could be seen as dismissive.

A message like "You don't know what you're talking about" fails at medium and high sensitivity because it attacks the person's competence directly. It might pass at low sensitivity since it doesn't contain explicit threats or hate speech, though it's borderline.

Content like "You're an idiot" or "People like you are the problem" fails at all sensitivity levels. These are clear personal attacks with no constructive value.

Finally, explicit threats like "I will find you and hurt you" fail at all sensitivity levels with maximum confidence. This is unambiguous toxic content.

Implementation Patterns

Here's how you'd typically use toxic language detection in different parts of your application.

For input validation, you check user messages before sending them to your AI or displaying them to other users:

```
async function validateUserMessage(message: string): Promise<boolean> {
  const result = await abv.guardrails.toxicLanguage.validate(
    message,
    { sensitivity: "medium" }
  );

  if (result.status === "pass") {
    return true;
  }

  // Log the reason for monitoring, but don't expose it to the user
  console.log("Blocked message:", result.reason);
  return false;
}

// Usage in your message handler
if (await validateUserMessage(userInput)) {
  await processMessage(userInput);
} else {
  return { error: "Your message violates our community guidelines." };
}
```

```python
async def validate_user_message(message: str) -> bool:
    result = await abv.guardrails.toxic_language.validate_async(
        message,
        {"sensitivity": "medium"}
    )

    if result["status"] == "pass":
        return True

    # Log the reason for monitoring, but don't expose it to the user
    print(f"Blocked message: {result['reason']}")
    return False

# Usage in your message handler
if await validate_user_message(user_input):
    await process_message(user_input)
else:
    return {"error": "Your message violates our community guidelines."}
```

For output validation, you check AI-generated responses before showing them to users:

```typescript
async function generateSafeResponse(prompt: string): Promise<string> {
  // Generate initial response
  let response = await callAI(prompt);

  // Validate the response
  const validation = await abv.guardrails.toxicLanguage.validate(
    response,
    { sensitivity: "high" }
  );

  // If toxic, regenerate with explicit safety instruction
  if (validation.status === "fail") {
    response = await callAI(
      prompt + "\n\nIMPORTANT: Respond in a professional, respectful tone."
    );
  }

  return response;
}
```

```python
async def generate_safe_response(prompt: str) -> str:
    # Generate initial response
    response = await call_ai(prompt)

    # Validate the response
    validation = await abv.guardrails.toxic_language.validate_async(
        response,
        {"sensitivity": "high"}
    )

    # If toxic, regenerate with explicit safety instruction
    if validation["status"] == "fail":
        response = await call_ai(
            f"{prompt}\n\nIMPORTANT: Respond in a professional, respectful tone."
        )

    return response
```

For handling ambiguous cases, you might implement a review queue for unsure results:

```
async function handleUserContent(content: string) {
  const result = await abv.guardrails.toxicLanguage.validate(
    content,
    { sensitivity: "medium" }
  );

  if (result.status === "pass") {
    // Content is clearly acceptable
    await publishContent(content);
  } else if (result.status === "fail" && result.confidence > 0.8) {
    // High-confidence violation, auto-reject
    await rejectContent(content, "Community guidelines violation");
  } else {
    // Low confidence or unsure - flag for human review
    await flagForModeration(content, result);
  }
}
```

```
async def handle_user_content(content: str):
    result = await abv.guardrails.toxic_language.validate_async(
        content,
        {"sensitivity": "medium"}
    )

    if result["status"] == "pass":
        # Content is clearly acceptable
        await publish_content(content)
    elif result["status"] == "fail" and result["confidence"] > 0.8:
        # High-confidence violation, auto-reject
        await reject_content(content, "Community guidelines violation")
    else:
        # Low confidence or unsure - flag for human review
        await flag_for_moderation(content, result)
```

Performance Optimization

Since toxic language detection uses AI, it takes one to three seconds per check and consumes tokens. You can optimize performance by running a fast rule-based check first to catch obvious violations before making the expensive AI call.

```
async function efficientToxicCheck(text: string): Promise<boolean> {
  // Quick check for explicitly forbidden terms (under 10ms, free)
  const quickCheck = await abv.guardrails.containsString.validate(
    text,
    {
      strings: ["explicit-slur", "forbidden-term"],
      mode: "none",
    }
  );

  // If quick check fails, no need for expensive AI check
  if (quickCheck.status === "fail") {
    return false;
  }

  // Only run AI check if quick check passed
  const deepCheck = await abv.guardrails.toxicLanguage.validate(text);
  return deepCheck.status === "pass";
}
```

```python
async def efficient_toxic_check(text: str) -> bool:
    # Quick check for explicitly forbidden terms (under 10ms, free)
    quick_check = await abv.guardrails.contains_string.validate_async(
        text,
        {
            "strings": ["explicit-slur", "forbidden-term"],
            "mode": "none"
        }
    )

    # If quick check fails, no need for expensive AI check
    if quick_check["status"] == "fail":
        return False

    # Only run AI check if quick check passed
    deep_check = await abv.guardrails.toxic_language.validate_async(text)
    return deep_check["status"] == "pass"
```

Security Best Practices

Never expose the reason field to end users. The reason explains why content failed validation, and exposing this information helps bad actors learn how to evade your guardrails. Instead, use generic error messages while logging the detailed reason internally for monitoring and improvement.

```javascript
// Bad - exposes validation logic
if (result.status === "fail") {
  return { error: result.reason };  // Don't do this!
}

// Good - generic message, internal logging
if (result.status === "fail") {
  logger.info("Blocked toxic content", { reason: result.reason });
  return { error: "Your message violates our community guidelines." };
}
```

```
# Bad - exposes validation logic
if result["status"] == "fail":
    return {"error": result["reason"]}  # Don't do this!

# Good - generic message, internal logging
if result["status"] == "fail":
    logger.info(f"Blocked toxic content: {result['reason']}")
    return {"error": "Your message violates our community guidelines."}
```

Choosing the Right Sensitivity

Here's a decision framework for choosing sensitivity based on your application type. If you're building for children or vulnerable populations, always use high sensitivity. The potential harm from allowing toxic content through far outweighs the cost of false positives.

If you're building a customer-facing application like customer service, social media, or collaborative tools, medium sensitivity is usually appropriate. It catches clear violations while allowing professional disagreement.

If you're building for professional or technical audiences where robust debate is expected, consider low sensitivity. Technical forums, code review systems, and professional feedback tools benefit from allowing strong opinions.

You can also adjust sensitivity based on user context. Authenticated users with good history might get lower sensitivity while anonymous users get higher sensitivity. Users who identify as minors automatically get high sensitivity regardless of the default setting.

Next Steps

The toxic language guardrail is often used alongside other guardrails for comprehensive content validation. Consider combining it with biased language detection for a more complete content safety solution. You might also want to use contains string to quickly catch explicit forbidden terms before running the more expensive toxic language check.

For more detailed implementation guidance, see the best practices documentation which covers optimization strategies, error handling, and monitoring approaches.

```
asdasdasdasdasdasdasdasdasdsa
```

22.2. ASDASDSDA

ASDASDASDAS

ASDASDASDAS

> ASDSADASDSAD

22.3. Test numbering on images issue (cloned)

The Entity Bank Details (EBD) bundle integrates with the standard NetSuite Vendor Record to securely house all necessary information for submitting payments. Once installed, imported vendor and employee data are accessible via **Setup → Entity Bank Details → Overview**. Learn how to import, export, and update EBD for both vendors and employees in this guide.

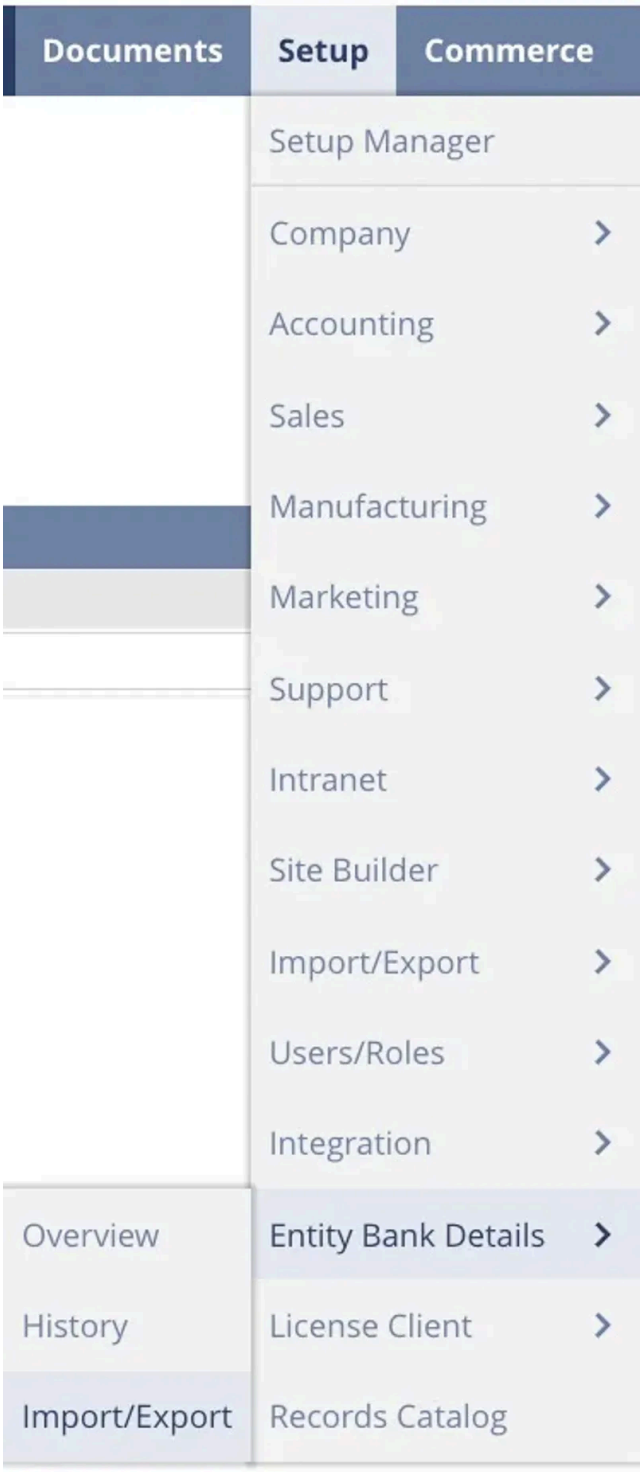
The process for importing and exporting records on the Vendor and Employee tabs is the same. The only difference is on the Vendor tab, which offers a specialized export utility for users with NetSuite's Electronic Bank Payments bundle, allowing them to conveniently transfer and re-upload data into EBD.

Import Entity Bank Data

The process for importing EBD data is the same for both Vendor Records and Employee Records.

1 Go to the Import/Export Tab

Go to **Setup → Entity Bank Details → Import/Export**



Go to Import/Export

2 Import CSV template

Select **Template** to download a blank CSV template. Use this CSV to fill out Bank Details for bulk upload.

Import Data section. Click on the Download Template button to get a blank CSV

> **NOTE:** Use this feature only to create new records. To update existing records, you must use the **Export CSV** feature, as it includes the necessary ID column, which is missing from the blank CSV template.

3 Format and Fill Out CSV

Please see the **Field Requirements for CSV** section of this page for a table containing all formatting requirements for the CSV file.

> **NOTE:** Confirm that you use the IBAN country code (e.g., CA).

> **TIP:** Have leading zeroes in your data? Apostrophes ['] in front of any leading zeroes or formatting cells as "plain text" will keep leading zeroes in Microsoft Excel or Google Sheets, but NOT Apple Numbers.

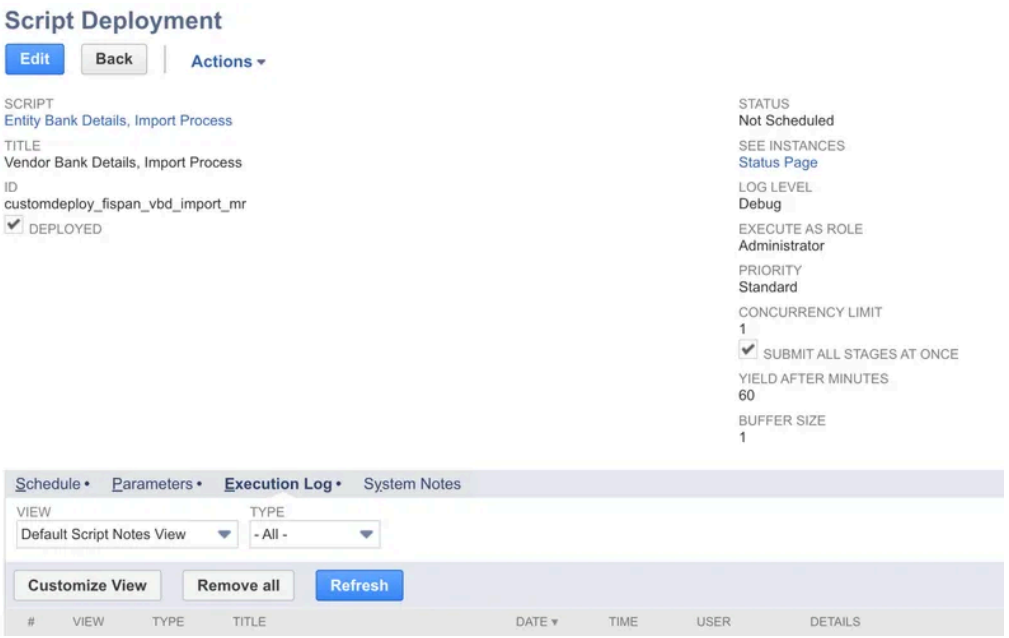
4 Upload the Formatted CSV

In the same Import/Export screen, upload the formatted CSV document.

Confirm successful upload by checking the **Import Logs** to ensure all records were uploaded successfully.

Import Data section

Select **Import logs** → **Execution logs** to confirm import completion.



Execution Logs

5 Check Information

Prior to sending payments, check the **Entity Bank Details** tab of the Vendor Record to ensure all information is correctly updated.

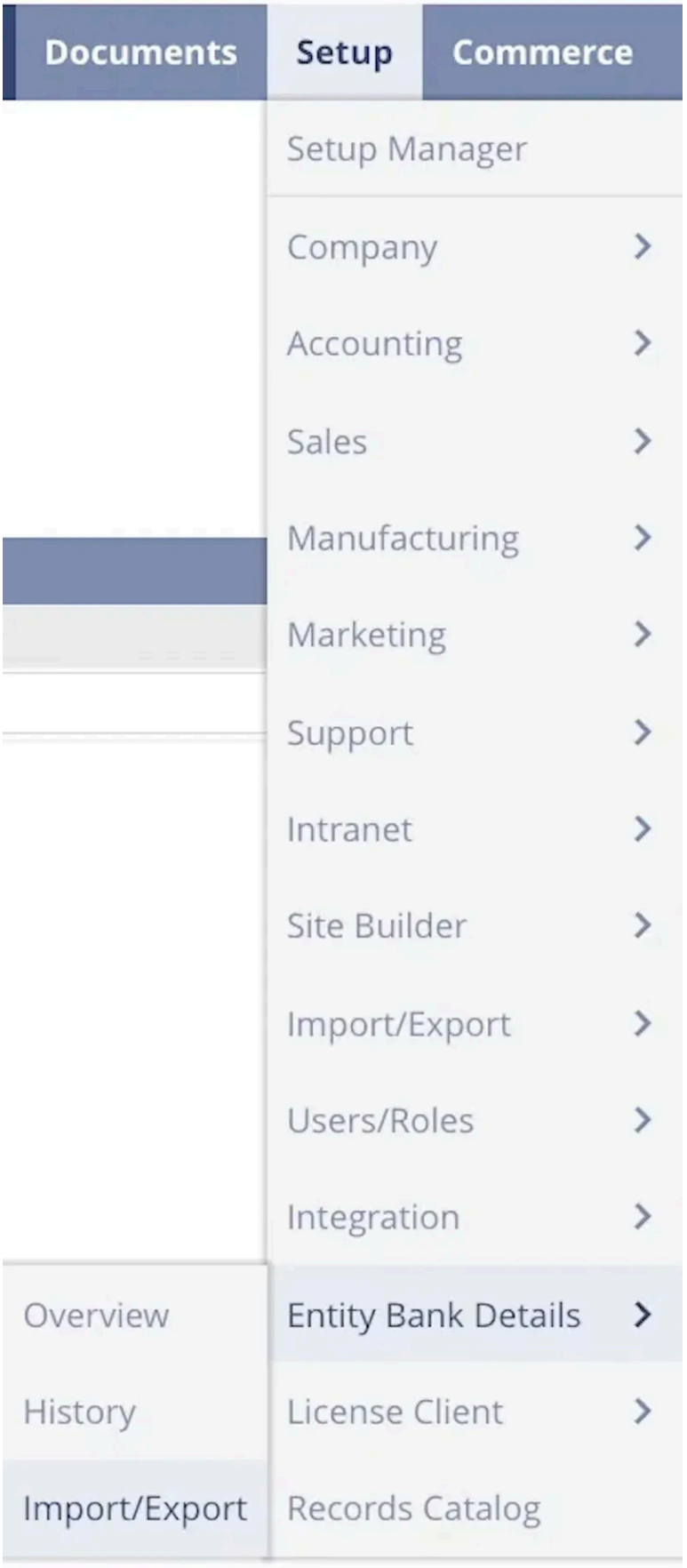
> **IMPORTANT:** All Entity Bank Details for both Vendors and Employees, alongside all History logs, are stored in your NetSuite environment. Please ensure you are regularly exporting your Entity Bank Detail records and your History logs to prevent any potential loss, such as through uninstallation of the bundle.

Export Entity Bank Data

The process for exporting EBD data is the same for both Vendor Records and Employee Records.

1 Go to the Import/Export Tab

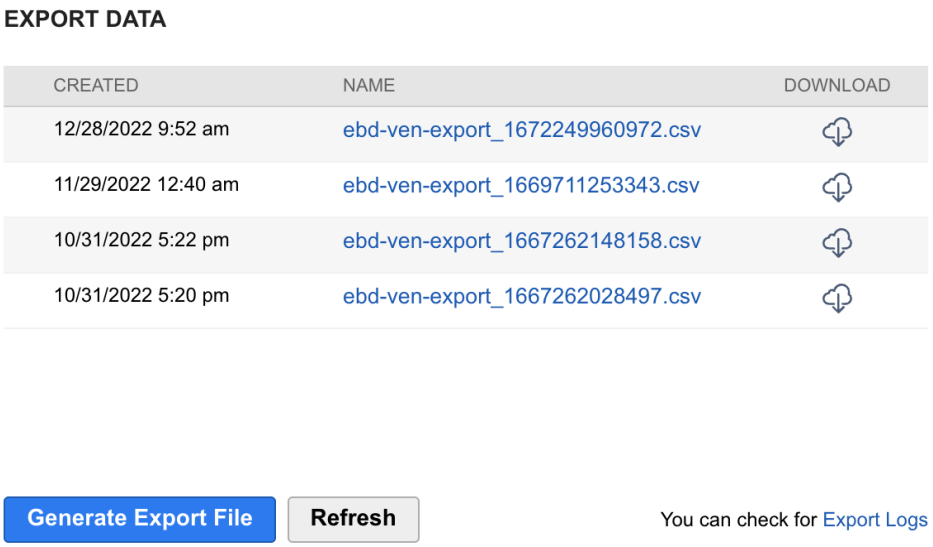
Go to **Setup** → **Entity Bank Details** → **Import/Export**



Go to Import/Export

2 Export CSV template

Click on the **Generate Export File** button to trigger a CSV download of all records kept in EBD.



Export Data section of the Import/Export page

The table will display a history of all exports created, alongside timestamps and buttons to download each of these files.

Export Logs contains more information about the export files generated. This can be found under the **Execution Log** tab. For each line item, it will report who initiated the export, the number of records exported in the one file, as well as the file name for that particular export.

Update Existing EBD Records

Use the Import/Export feature to bulk edit existing Entity Bank Details (EBD) for multiple Employee and Vendor Records at once.

1 Export CSV File

To update existing Entity Bank Details (EBD) records, you must first export a CSV file of those records.

The exported CSV file is crucial because it contains a unique ID column. This ID column is not present in the blank CSV import template and is necessary for the system to identify and update existing data.

2 Make edits to the CSV

Using your preferred application (such as Microsoft Excel or Google Sheets), open the exported CSV file to make your edits.

ebd-ven-export_

	id	vendor	label	primary	country	currency	method	bic	accountType	accountNumber	routingNumber
1	1	1305	US-DOMESTIC-USD	T	US	USD	DOMESTIC		CHECKING		

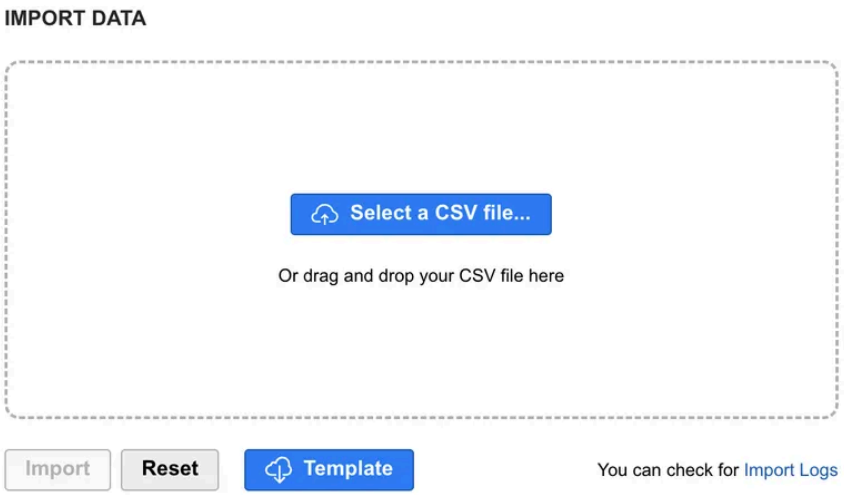
EBD Vendor Export CSV opened in Numbers. Note the ID column on the far left

Ensure the **ID** column on the far left remains untouched.

You can also add new records to the list; simply enter them as a new row at the bottom of the CSV file. The system will assign an ID to these new records upon import.

3 Import the edited Export file

After making edits to the previously exported file, you may now import the CSV file back into the Entity Bank Details bundle.



Import Data section. Click "Select a CSV file" or drag-and-drop a CSV file in this section to import your CSV file

Either **Select a CSV file** to upload, or drag and drop one into the **Import Data** section of the Import/Export page.

Troubleshooting Common Import Issues

Issue	Resolution
No success message appears in NetSuite after importing Entity Bank Data.	Check the import logs directly to verify the import status.
Records were successfully created but appear incomplete or inaccurate.	Records can be created even if they are missing required fields or have incorrect formatting. Validate the imported records for accuracy after every upload.
Duplicate bank records are created for a vendor/employee.	This happens if the same records are imported more than once. Ensure you are importing unique vendor or employee bank accounts with each import file.
The international payment method is incorrectly set or missing.	When importing, the payment method for all international wire accounts must be entered as INTERNATIONAL_WIRE in the EBD template. Additionally, ensure bank accounts for international wire countries are not set to the DOMESTIC payment method in the vendor's NetSuite profile.

FAQs

Is it possible to add additional fields?

No. NetSuite does not currently allow the addition of fields to the EBD record.

Why are the Bank Balance and Balance As Of fields in the Match Bank Data page blank after a successful import?

This is a NetSuite limitation. The system does not populate bank balance information because it relies on the usage of a bank feed, which is not integrated here.

Field Requirements for CSV

Make sure your CSV Import file follows these requirements.

IMPORTANT: Ensure any data with leading zeroes is preceded by an apostrophe `'` or formatted as plain text in Microsoft Excel or Google Sheets.

NOTE: If you are using payment methods that require mandatory country codes for Entity Bank Details, there may be additional columns that must be included in the import.

To ensure a successful import, we recommend first manually adding the Entity Bank Details to the ERP system. Then, download the template; it will already display the correct columns needed for entering any remaining details.

Name	Values	Description
id	Numeric	This is the ID number for this specific record. This column is not available on the Import Template as it is system-generated.
vendor (or employee)	NetSuite Vendor ID or Employee ID	Required. The Vendor or Employee ID can be found in the URL of the entity record or on the EBD Overview page.
label	Anything	Required. This can be entity or bank details. This field is used to differentiate the records when doing mass uploads.
primary	YES/NO, Y/N, TRUE/FALSE, T/F	Required. T = True, F = False. If there are multiple bank details for a single payment method on a single vendor being imported, mark one as T and all others F.
country	2-letter ISO country code (e.g., US, CA, UK)	Required. Field accepts upper, lower, and mixed case values.
currency	3-letter currency code required (e.g., USD, CAD)	Required. Field accepts upper, lower, and mixed case values.
method	DOMESTIC, INTERNATIONAL_WIRE	Required. Has to be in the same format as it is case sensitive. Please use INTERNATIONAL_WIRE for bank accounts set up for international wire payments. Please use DOMESTIC for US/Canada-based accounts.
bankName	Anything	
address.line1	Anything	
address.city	Anything	

Name	Values	Description
address.stateProvince	Anything	
address.postalCode	Anything	
bic	Valid BIC or SWIFT code for the country and bank	
accountType	CHECKING, SAVINGS	Field accepts upper, lower, and mixed case values.
accountNumber	Valid account number for this bank	
localBranchCode	Valid branch code for this bank	Use this field for BSB codes if the country requires it.
paymentDefaults.purposeMessage	Anything	
paymentDefaults.purposeCode	Valid purpose code	
paymentDefaults.paymentIsoCode	Valid ISO code	
paymentDefaults.paymentCodeword	Valid payment code	
paymentDefaults.pay	Valid party type	P = Parent, T = Subsidiary, G = Group, N = Non-related

Name	Values	Description
mentPartyType		
paymentDefaults.residentialStatus	Valid residential status	
bankCode	Valid bank code for this bank	
iban	2-letter country code, followed by two check digits, and up to 35 alphanumeric characters	
sortCode	Valid sort code for this bank	
institutionNumber	Valid institution number for this bank	
transitNumber	Valid transit number for this bank	
routingNumber	Valid routing number for this bank	

Transferring Electronic Bank Payments Data

Before the Entity Bank Details (EBD) bundle is installed, you are limited to making only ACH and CPA payments. This is done through the bank plugin, using vendor information stored in NetSuite's Electronic Bank Payments (EBP) bundle.

Once EBD is installed, the EBP bundle is no longer used to pull vendor banking data for bill payments. Instead, EBD becomes the new source of payment information.

The EBD bundle provides the ability to transfer ACH and CPA vendor banking information from the EBP bundle via CSV.

ACH and CPA banking information stored in the following default Payment File Formats are supported:

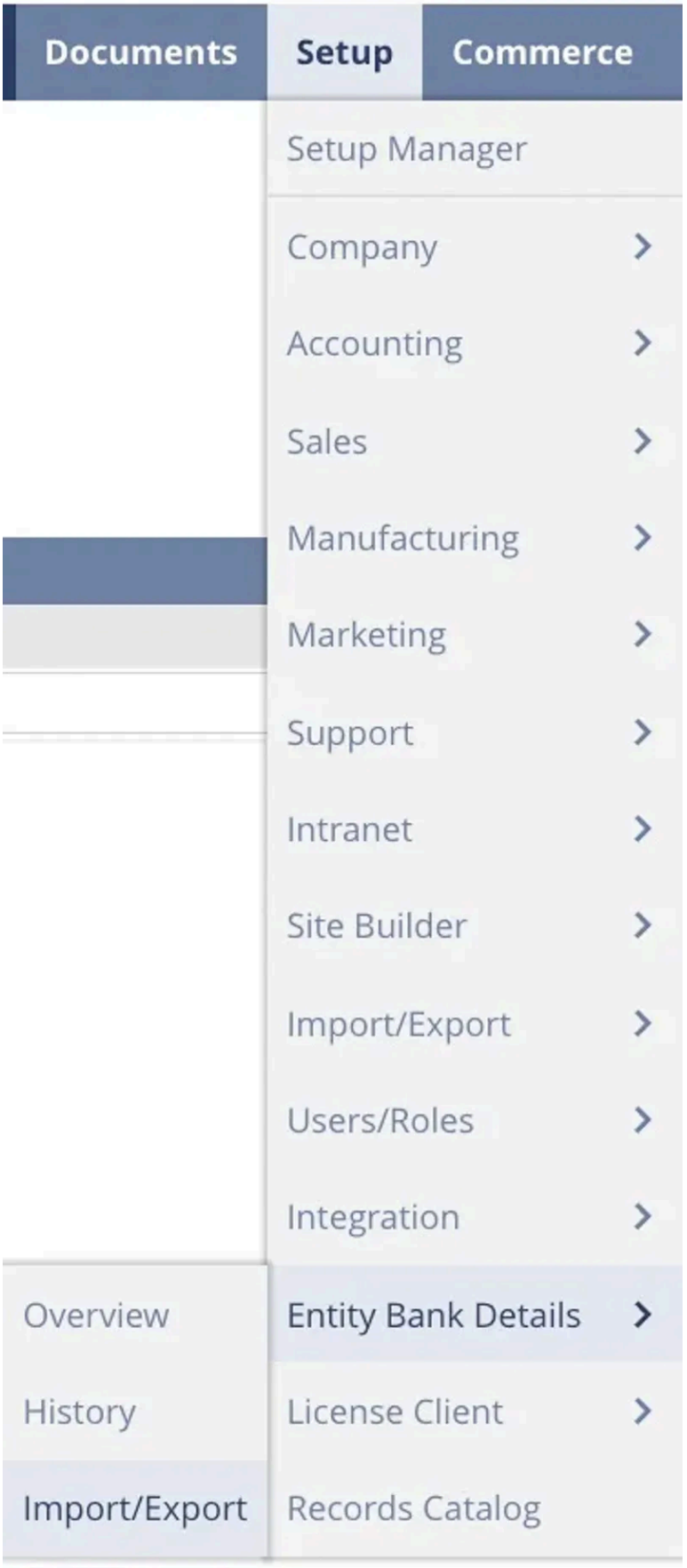
- ACH - CCD/PPD
- ACH - CTX (Free Text)
- CPA-005

Transferring ACH and CPA banking information that is stored in custom Payment File Formats is unsupported.

To quickly transfer your supported data across bundles, follow the steps below.

1 Go to the Import/Export Tab

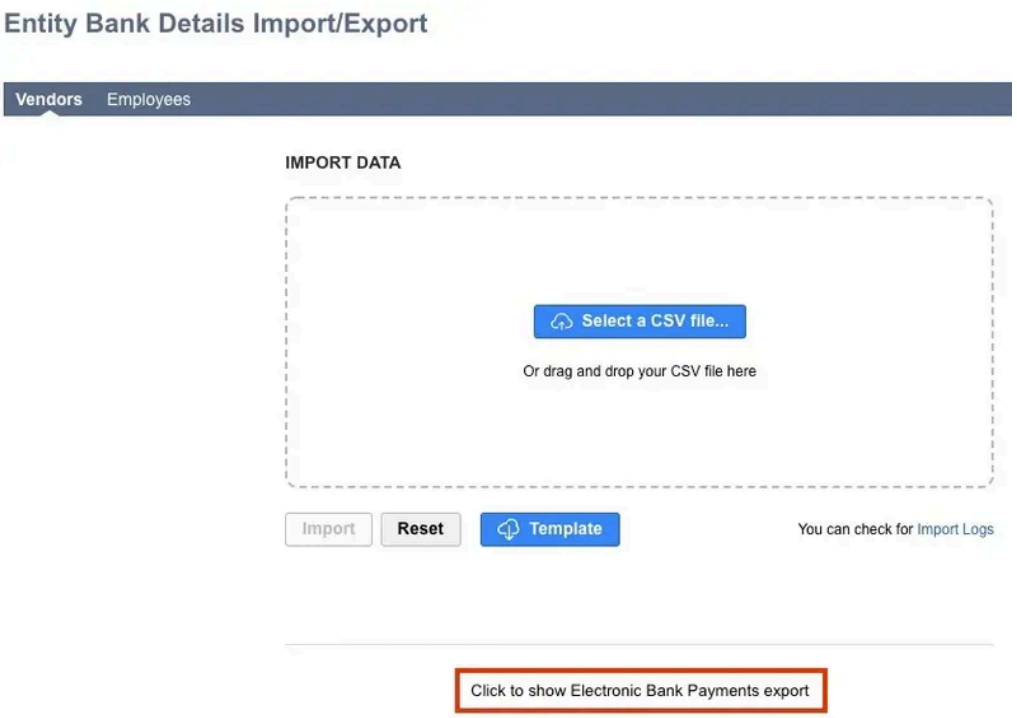
Go to **Set Up** → **Entity Bank Details** → **Import/Export**



Go to Import/Export

2 Generate CSV(s)

Select **Click to show Electronic Bank Payments export**

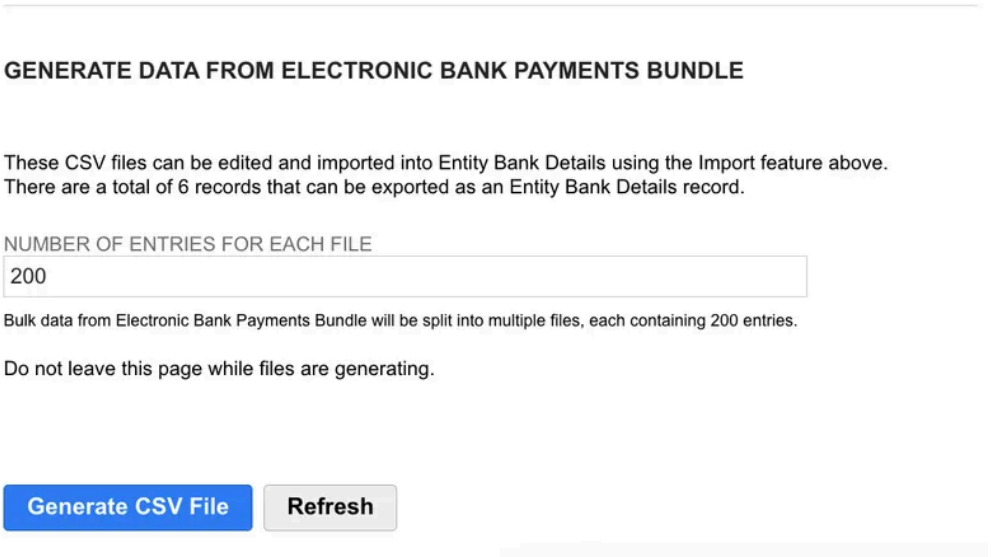


Click to show Electronic Bank Payments export

Select **Generate CSV File.**

This will generate a CSV file(s) dependent on the **Number of Entries for Each File**, set below.

NetSuite limits the number of records that can be pulled per file. For example, if you have 1100 records and set the number of entries to 200, the bundle will generate five CSVs containing 200 records and one CSV containing 100 records.



Generate CSV file

> **NOTE:** The **Generate Data from Electronic Bank Payments Bundle** can only extract ACH and CPA payment details. Any additional payment details must be transferred into the Entity Bank Details bundle either via manual entry or using the Import feature.

3 Import CSV(s)

Select a CSV file... or drag and drop the CSV file(s) into the **Import Data** box.

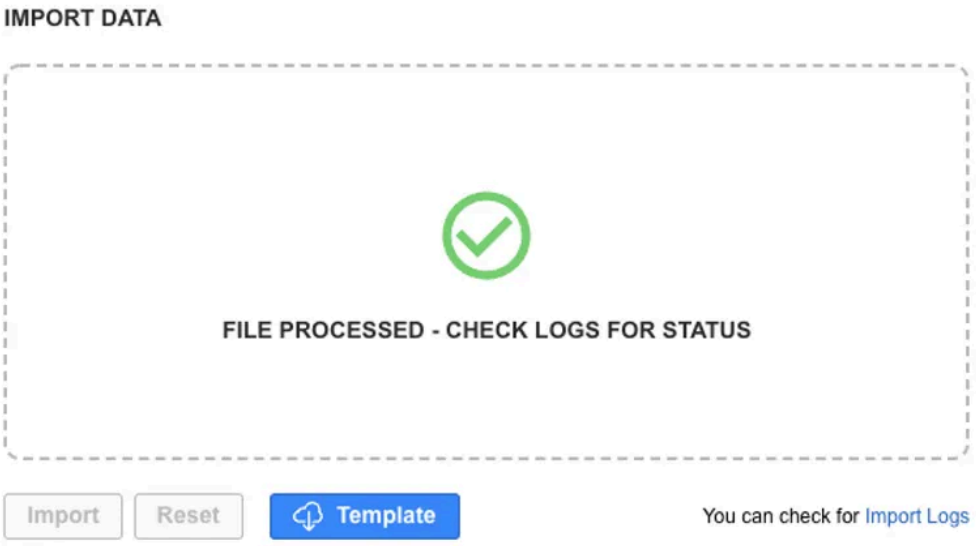
Import Data section

Then, select **Import.**

Wait a few moments for the import to complete.

4 Confirm Import Completion

Once complete, select **Import logs** → **Execution logs** to confirm import completion.



Confirm Status

You can also check the following places to confirm bank details have been saved:

- Entity Bank Details tab on the Vendor or Employee Record
- Entity Bank Details Overview page
- Entity Bank Details History page

22.3.1. asdsadas

About Profile | Frequently Asked Questions

Profile at a Glance

Whether you're just getting started or looking for quick tips, this section will guide you through the basics of using Profile. Each article includes links to related topics, so you can explore more when needed.

This section introduces the core features and functions of Profile that support day-to-day use. It covers general information such as logging in and out, locking your screen, navigating between windows, and customising your workspace. You'll also find guidance on using filters and search tools to locate information, shortcut keys, layout settings to improve efficiency, and more.

These resources are designed to help all users build confidence and stay productive when working in Profile.

Get Started

If you would like to know more about general information related to Profile please contact your Customer Success Manager.

Job Aids

Keep key information within reach. Click below to download printable quick-reference guides:

JobAid-IntroductionToProfile.pdf

Related Articles

Read the related articles below for more general information about Profile:

- **Logging into Profile**
- **Logging Out of Profile**
- **Profile Navigation Overview**
- **Using Search Across Profile**
- **Navigate the Work Centre Window**
- **Navigate Medical Records in Profile**
- **Work with Profile Windows**
- **Use the Clinical Details Window**
- **Use Segment Lists and Record Views**
- **Insert and Edit Information in Profile**
- **Use Filters Across Profile**
- **Changing Place of Service**
- **Logging Out of Profile**
- **Lock your Screen in Profile**
- **Save your Profile Windows Settings**
- **Switch Between Open Windows in Profile**
- **Using Shortcut Keys**

22.3.2. Toxic Language

The toxic language guardrail detects harmful content including hate speech, threats, insults, and other communication that could damage your community or brand. Unlike simple keyword filters, this guardrail uses AI to understand context, tone, and intent.

When to Use This Guardrail

You should use toxic language detection when you need to protect your community from harmful interactions, prevent your AI from receiving poisoned context that could influence its behavior, or maintain brand safety in AI-generated outputs. This guardrail is particularly valuable in applications with user-generated content like forums, chat systems, and comment sections, as well as in customer service scenarios where you need to catch hostile messages before they reach agents or AI systems.

The key advantage over simple keyword filtering is that this guardrail understands nuance. Someone can be hostile without using profanity, and they can use strong language without being hostile. The guardrail analyzes intent and tone, not just word choice.

Understanding Sensitivity Levels

The sensitivity setting controls how strict the guardrail is. Think of it as adjusting the threshold for what constitutes a violation. This is one of the most important configuration choices you'll make because it fundamentally changes what content passes or fails.

Low sensitivity flags only severe violations like explicit threats and extreme hate speech. When you set sensitivity to low, you're saying that you expect strong opinions and robust disagreement, and you only want to block content that crosses a clear line into threatening or hateful territory. For example, "I strongly disagree with this approach and think it's misguided" would pass at low sensitivity, as would "This is a terrible idea." Only content like "I will find you and hurt you" or explicit hate speech would fail.

Low sensitivity works well for public forums where debate is expected, professional feedback environments where directness is valued, and technical communities where people discuss contentious topics. The tradeoff is that some content that makes people uncomfortable might still pass through.

Medium sensitivity is the default setting and represents a balanced approach. At this level, the guardrail flags clear violations including insults and hostile language while still allowing professional disagreement and constructive criticism. A message like "I disagree with your reasoning" would pass, but "You're an idiot" or "People like you are the problem" would fail.

Medium sensitivity works well for most applications including customer service systems, business communications, collaborative tools, and social platforms. It strikes a balance between allowing meaningful discourse and maintaining a respectful environment.

High sensitivity creates the strictest environment by flagging any potentially toxic content including mild rudeness and dismissive language. At this level, even content like "Whatever, dude" or "That's pretty dumb" would fail. Only respectful, neutral content passes at high sensitivity.

High sensitivity is appropriate for children's applications where you need maximum protection, educational platforms where you want to model respectful communication, safe spaces and support communities where people need to feel secure, and compliance-critical contexts where any potential issue needs to be caught.

Configuration Options

The toxic language guardrail accepts several configuration options that let you tune its behavior for your specific use case.

```
// Available options:
// sensitivity: "low", "medium", or "high" (default: "medium")
// model: model identifier (default: Claude 3.5 Haiku)
// temperature: 0-1, lower values = more consistent (default: 0.1)
// maxTokens: response length limit (default: 200)

await abv.guardrails.toxicLanguage.validate(text, {
  sensitivity: "medium",
  model: "model-name",
  temperature: 0.1,
  maxTokens: 200,
});
```

```
# Available options:
# sensitivity: "low", "medium", or "high" (default: "medium")
# model: model identifier (default: Claude 3.5 Haiku)
# temperature: 0-1, lower values = more consistent (default: 0.1)
# maxTokens: response length limit (default: 200)

abv.guardrails.toxic_language.validate(text, {
    "sensitivity": "medium",
    "model": "model-name",
    "temperature": 0.1,
    "maxTokens": 200
})
```

The sensitivity option is the most important and you'll use it frequently. The model, temperature, and maxTokens options are advanced settings that you typically won't need to change. The default model is optimized for guardrail tasks and provides the best balance of speed, accuracy, and cost. The default temperature of 0.1 ensures consistent results. The default maxTokens of 200 is sufficient for the explanation field.

Real-World Examples

Let's look at concrete examples of how different sensitivity levels handle various types of content. Understanding these patterns will help you choose the right sensitivity for your application.

Consider a message like "I disagree with your approach to this problem." This is professional disagreement and passes at all sensitivity levels. The language is neutral and respectful despite expressing disagreement.

Now consider "This is a terrible idea and shows poor judgment." This passes at low and medium sensitivity because while it's critical, it focuses on the idea rather than attacking the person. However, it might fail at high sensitivity because "terrible" and "poor judgment" could be seen as dismissive.

A message like "You don't know what you're talking about" fails at medium and high sensitivity because it attacks the person's competence directly. It might pass at low sensitivity since it doesn't contain explicit threats or hate speech, though it's borderline.

Content like "You're an idiot" or "People like you are the problem" fails at all sensitivity levels. These are clear personal attacks with no constructive value.

Finally, explicit threats like "I will find you and hurt you" fail at all sensitivity levels with maximum confidence. This is unambiguous toxic content.

Implementation Patterns

Here's how you'd typically use toxic language detection in different parts of your application.

For input validation, you check user messages before sending them to your AI or displaying them to other users:

```
async function validateUserMessage(message: string): Promise<boolean> {
  const result = await abv.guardrails.toxicLanguage.validate(
    message,
    { sensitivity: "medium" }
  );

  if (result.status === "pass") {
    return true;
  }

  // Log the reason for monitoring, but don't expose it to the user
  console.log("Blocked message:", result.reason);
  return false;
}

// Usage in your message handler
if (await validateUserMessage(userInput)) {
  await processMessage(userInput);
} else {
  return { error: "Your message violates our community guidelines." };
}
```

```python
async def validate_user_message(message: str) -> bool:
    result = await abv.guardrails.toxic_language.validate_async(
        message,
        {"sensitivity": "medium"}
    )

    if result["status"] == "pass":
        return True

    # Log the reason for monitoring, but don't expose it to the user
    print(f"Blocked message: {result['reason']}")
    return False

# Usage in your message handler
if await validate_user_message(user_input):
    await process_message(user_input)
else:
    return {"error": "Your message violates our community guidelines."}
```

For output validation, you check AI-generated responses before showing them to users:

```typescript
async function generateSafeResponse(prompt: string): Promise<string> {
  // Generate initial response
  let response = await callAI(prompt);

  // Validate the response
  const validation = await abv.guardrails.toxicLanguage.validate(
    response,
    { sensitivity: "high" }
  );

  // If toxic, regenerate with explicit safety instruction
  if (validation.status === "fail") {
    response = await callAI(
      prompt + "\n\nIMPORTANT: Respond in a professional, respectful tone."
    );
  }

  return response;
}
```

```python
async def generate_safe_response(prompt: str) -> str:
    # Generate initial response
    response = await call_ai(prompt)

    # Validate the response
    validation = await abv.guardrails.toxic_language.validate_async(
        response,
        {"sensitivity": "high"}
    )

    # If toxic, regenerate with explicit safety instruction
    if validation["status"] == "fail":
        response = await call_ai(
            f"{prompt}\n\nIMPORTANT: Respond in a professional, respectful tone."
        )

    return response
```

For handling ambiguous cases, you might implement a review queue for unsure results:

```typescript
async function handleUserContent(content: string) {
  const result = await abv.guardrails.toxicLanguage.validate(
    content,
    { sensitivity: "medium" }
  );

  if (result.status === "pass") {
    // Content is clearly acceptable
    await publishContent(content);
  } else if (result.status === "fail" && result.confidence > 0.8) {
    // High-confidence violation, auto-reject
    await rejectContent(content, "Community guidelines violation");
  } else {
    // Low confidence or unsure - flag for human review
    await flagForModeration(content, result);
  }
}
```

```python
async def handle_user_content(content: str):
    result = await abv.guardrails.toxic_language.validate_async(
        content,
        {"sensitivity": "medium"}
    )

    if result["status"] == "pass":
        # Content is clearly acceptable
        await publish_content(content)
    elif result["status"] == "fail" and result["confidence"] > 0.8:
        # High-confidence violation, auto-reject
        await reject_content(content, "Community guidelines violation")
    else:
        # Low confidence or unsure - flag for human review
        await flag_for_moderation(content, result)
```

Performance Optimization

Since toxic language detection uses AI, it takes one to three seconds per check and consumes tokens. You can optimize performance by running a fast rule-based check first to catch obvious violations before making the expensive AI call.

```
async function efficientToxicCheck(text: string): Promise<boolean> {
  // Quick check for explicitly forbidden terms (under 10ms, free)
  const quickCheck = await abv.guardrails.containsString.validate(
    text,
    {
      strings: ["explicit-slur", "forbidden-term"],
      mode: "none",
    }
  );

  // If quick check fails, no need for expensive AI check
  if (quickCheck.status === "fail") {
    return false;
  }

  // Only run AI check if quick check passed
  const deepCheck = await abv.guardrails.toxicLanguage.validate(text);
  return deepCheck.status === "pass";
}
```

```python
async def efficient_toxic_check(text: str) -> bool:
    # Quick check for explicitly forbidden terms (under 10ms, free)
    quick_check = await abv.guardrails.contains_string.validate_async(
        text,
        {
            "strings": ["explicit-slur", "forbidden-term"],
            "mode": "none"
        }
    )

    # If quick check fails, no need for expensive AI check
    if quick_check["status"] == "fail":
        return False

    # Only run AI check if quick check passed
    deep_check = await abv.guardrails.toxic_language.validate_async(text)
    return deep_check["status"] == "pass"
```

Security Best Practices

Never expose the reason field to end users. The reason explains why content failed validation, and exposing this information helps bad actors learn how to evade your guardrails. Instead, use generic error messages while logging the detailed reason internally for monitoring and improvement.

```javascript
// Bad - exposes validation logic
if (result.status === "fail") {
  return { error: result.reason };  // Don't do this!
}

// Good - generic message, internal logging
if (result.status === "fail") {
  logger.info("Blocked toxic content", { reason: result.reason });
  return { error: "Your message violates our community guidelines." };
}
```

```
# Bad - exposes validation logic
if result["status"] == "fail":
    return {"error": result["reason"]}  # Don't do this!

# Good - generic message, internal logging
if result["status"] == "fail":
    logger.info(f"Blocked toxic content: {result['reason']}")
    return {"error": "Your message violates our community guidelines."}
```

Choosing the Right Sensitivity

Here's a decision framework for choosing sensitivity based on your application type. If you're building for children or vulnerable populations, always use high sensitivity. The potential harm from allowing toxic content through far outweighs the cost of false positives.

If you're building a customer-facing application like customer service, social media, or collaborative tools, medium sensitivity is usually appropriate. It catches clear violations while allowing professional disagreement.

If you're building for professional or technical audiences where robust debate is expected, consider low sensitivity. Technical forums, code review systems, and professional feedback tools benefit from allowing strong opinions.

You can also adjust sensitivity based on user context. Authenticated users with good history might get lower sensitivity while anonymous users get higher sensitivity. Users who identify as minors automatically get high sensitivity regardless of the default setting.

Next Steps

The toxic language guardrail is often used alongside other guardrails for comprehensive content validation. Consider combining it with biased language detection for a more complete content safety solution. You might also want to use contains string to quickly catch explicit forbidden terms before running the more expensive toxic language check.

For more detailed implementation guidance, see the best practices documentation which covers optimization strategies, error handling, and monitoring approaches.

22.4. asdsadasdas (cloned)

Get Cakes

Try it

Get a cake by its ID

GET https://api.cakes.com

Request

BODY PARAMETERS

id `String` `required`

ID of the cake to get

Parameter name ▶ `Object` `optional`

Parameter name ▶ `Object` `optional`

```
var myHeaders = new Headers();
myHeaders.append("Accept", "application/json");
myHeaders.append("Content-Type", "application/json");

var raw = JSON.stringify({
   "id": "String"
});

var requestOptions = {
   method: 'GET',
   headers: myHeaders,
   body: raw,
   redirect: 'follow'
};

fetch("https://api.cakes.com", requestOptions)
   .then(response => response.text())
   .then(result => console.log(result))
   .catch(error => console.log('error', error));
```

```
{
  "name": "Cake's name",
}
```

23. Virtual Account Number API Specification

23.1. Create an Automatcher (Receivables) Account

POST https://api.mpay.com.au/receivables/v1/create Try it

Production URL

Creates a new virtual account number. These account numbers can be used along with the returned BSB to receive payments from any bank in Australia.

Payments to an invalid bank-account number, i.e. one that has not been created, will be returned by Monoova to the remitting institution. Each account number has a check digit to minimize common payer errors such as adjacent transposition errors or single-digit typos.

An optional unique ID can be linked to each account number for your reference. If a unique ID is not provided, Monoova will generate and return one automatically.

If isActive is omitted, it will default to false. This can be changed to true via the /receivables/v1/status endpoint.

For the creation of batches of account numbers, please see /receivables/v1/batchCreate.

Body Parameters

body `automatcherCreateBody_V1` *

Responses

200 successful validation

automatcherStatusResponse_V1

```
curl --request POST \
     --url https://api.mpay.com.au/receivables/v1/create \
     --header 'accept: application/json'
```

```
// successful validation
{}
```

23.2. Get Account Status By Bank Account

GET https://api.mpay.com.au/receivables/v1/statusByBank... Try it

Production URL

This API is used for checking the status of a bank account using the bank account number. This API will only return statuses for virtual accounts belonging to the parent mAccount.

Path Parameters

bankAccountNumber `string` *

Query Parameters

bsb `string`

Responses

200 successful validation

`automatcherStatusResponse_V1`

```
curl --request GET \
     --url https://api.mpay.com.au/receivables/v1/statusByBankAccount/{bankAccountNumber}
     --header 'accept: application/json'
```

```
// successful validation
{}
```

23.3. Batch Create

POST https://api.mpay.com.au/receivables/v1/batchCreate Try it

Production URL

The batch create API has the same functionality as the /receivables/v1/create endpoint however it allows you to create up to 50,000 accounts in a single call by submitting a file. We recommend this for circumstances where more than 100 accounts need to be created within a short time period.

Creation of accounts with isActive=true is restricted to 1000. Multiple batches cannot be requested at the same time, you have to wait for the completion of a batch before requesting a new one.

The Batch Create API also supports a lightweight option whereby many accounts can be created by submitting a single line. This option automatically sets isActive = true, hence you can only request the creation of 1000 accounts in one request. For this method, all accounts will be created with the same specified name, and clientuniqueidâ€™s cannot be specified will be automatically generated.

Responses

● 200 successful validation

string<binary>

```
curl --request POST \
     --url https://api.mpay.com.au/receivables/v1/batchCreate \
     --header 'accept: application/json'
```

```
// successful validation
""
```

23.4. Set Account Status

POST https://api.mpay.com.au/receivables/v1/status Try it

Production URL

Update the status of a virtual account. Accounts can be active or inactive. If an account is inactive, all inbound payments will be rejected and returned to the payer.

Body Parameters

body `automatcherStatusBody_V1` *

Responses

● 200 successful validation

`automatcherStatusResponse_V1`

```
curl --request POST \
     --url https://api.mpay.com.au/receivables/v1/status \
     --header 'accept: application/json'
```

```
// successful validation
{}
```

23.5. Batch Status

POST https://api.mpay.com.au/receivables/v1/batchStatus Try it

Production URL

Get the status of a request to the batchCreate endpoint.

Responses

200 successful validation

string<binary>

```
curl --request POST \
     --url https://api.mpay.com.au/receivables/v1/batchStatus \
     --header 'accept: application/json'
```

```
// successful validation
""
```

23.6. Get Account Status By ClientUniqueId

GET https://api.mpay.com.au/receivables/v1/statusByClien... Try it

Production URL

This API is used for checking the status of a bank account using the clientUniqueId provided when the account was created.

Path Parameters

> **clientUniqueId** `string` *

Query Parameters

bsb `string`

Responses

● 200 successful validation

`automatcherStatusResponse_V1`

```
curl --request GET \
     --url https://api.mpay.com.au/receivables/v1/statusByClientID/{clientUniqueId} \
     --header 'accept: application/json'
```

```
// successful validation
{}
```

23.7. securitySchemes

BASICAUTH | https://api.mpay.com.ausecuritySchemes | Try it

Production URL

Responses

- 200

```
curl --request BASICAUTH \
     --url https://api.mpay.com.ausecuritySchemes \
     --header 'accept: application/json'
```

23.8. schemas

AUTOMATCHERCREATEBODY_V1 | https://api.mpay.com.auschemas | Try it

Production URL

Responses

200

```
curl --request AUTOMATCHERCREATEBODY_V1 \
     --url https://api.mpay.com.auschemas \
     --header 'accept: application/json'
```

23.9. schemas

AUTOMATCHERBATCHDETAILSCLASS | https://api.mpay.com.aus... | Try it

Production URL

Responses

200

```
curl --request AUTOMATCHERBATCHDETAILSCLASS \
     --url https://api.mpay.com.auschemas \
     --header 'accept: application/json'
```

23.10. schemas

AUTOMATCHERBATCHRESPONSE_V1 | https://api.mpay.com.ausc... | Try it

Production URL

Responses

200

```
curl --request AUTOMATCHERBATCHRESPONSE_V1 \
     --url https://api.mpay.com.auschemas \
     --header 'accept: application/json'
```

23.11. schemas

DEBUGRESPONSE | https://api.mpay.com.auschemas | Try it

Production URL

Responses

200

```
curl --request DEBUGRESPONSE \
     --url https://api.mpay.com.auschemas \
     --header 'accept: application/json'
```

23.12. schemas

AUTOMATCHERSTATUSBODY_V1 | https://api.mpay.com.auschemas | Try it

Production URL

Responses

200

```
curl --request AUTOMATCHERSTATUSBODY_V1 \
     --url https://api.mpay.com.auschemas \
     --header 'accept: application/json'
```

23.13. schemas

AUTOMATCHERSTATUSRESPONSE_V1 | https://api.mpay.com.aus... | Try it

Production URL

Responses

200

```
curl --request AUTOMATCHERSTATUSRESPONSE_V1 \
     --url https://api.mpay.com.auschemas \
     --header 'accept: application/json'
```

24. Judopay Transaction API

24.1. Registering Cards

24.1.1. /transactions/savecard

POST https://api-sandbox.judopay.com/transactions/savecard Try it

Sandbox environment

If you do not want to perform a pre-authorisation check on a customer's account, use this call to tokenize the card information into an encrypted string.

Note: The Card Token is not tested until it is used.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

> Api-Version string *

Body Parameters

body saveCardRequestPan *

All Of
object +

Description

Request is sent with full card details

> cardNumber string *

> expiryDate string *

> startDate string

> yourConsumerReference string *

> judoId string

> currency string

> cardHolderName string

> disableNetworkTokenisation boolean

> cardAddress cardAddressRequestAttributes

All Of
cardAddressCom...

Responses

200 Successful response

- transactionReceiptResponse
  - receiptId string
  - originalReceiptId string
  - yourPaymentReference string
  - type string<Payment | Refund | PreAuth…
  - createdAt string<date-time>
  - result string
  - message string
  - judoId integer
  - merchantName string
  - appearsOnStatementAs string
  - originalAmount string
  - amountCollected string
  - netAmount string
  - amount string
  - currency string
  - recurringPaymentType string
  - acquirerTransactionId string
  - externalBankResponseCode string
  - authCode string
  - walletType integer
  - riskScore integer

- paymentNetworkTransactionId `string`
- allowIncrement `boolean`
- isIncrementalAuth `boolean`
- emailAddress `string`
- cardDetails `cardDetailsResponse`
- billingAddress `cardAddressResponseAttribu…` All Of **cardAddressCom…**
- consumer `consumerResponse`
- device `deviceResponse`
- yourPaymentMetaData `object`
- threeDSecure `threeDSecureCompletedRespo…`
- risks `riskParametersResponse`
- disableNetworkTokenisation `boolean`
- networkTokenisationDetails `networkTokenisa…`

● **400** Bad request if any of the required request parameters are not supplied

● **401** Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

● **403** Forbidden if token used does not have required permission

- errorResponse
  - requestId string
  - message string
  - code integer
  - category integer

```
curl --request POST \
     --url https://api-sandbox.judopay.com/transactions/savecard \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "cardNumber": "1234567890123456",
     "expiryDate": "01/28",
     "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
     "cardAddress": {
     "postCode": "AB1 2CD"
     }
     }'
```

```
// Successful response
{
  "receiptId": "1001131610340495360",
  "originalReceiptId": "1001124307998347264",
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "type": "PreAuth",
  "createdAt": "2025-02-05T16:28:32.8596+00:00",
  "result": "Success",
  "message": "AuthCode: 123456",
  "judoId": 100100100,
  "merchantName": "Test Merchant",
  "appearsOnStatementAs": "APL*/TestMerchant       ",
  "originalAmount": "10.99",
  "amountCollected": "1.99",
  "netAmount": "1.00",
  "amount": "1.00",
  "currency": "GBP",
  "recurringPaymentType": "MIT",
  "acquirerTransactionId": "33666277256892153705",
  "externalBankResponseCode": "A",
  "authCode": "123456",
  "walletType": 1,
  "riskScore": 0,
  "paymentNetworkTransactionId": "123456789012345",
  "allowIncrement": true,
  "isIncrementalAuth": true,
```

24.2. Pre-Authorization

24.2.1. /transactions/checkcard

POST https://api-sandbox.judopay.com/transactions/checkcard Try it

Sandbox environment

Conduct a zero amount pre-authorisation in the customer's account. Use this endpoint to check if the card is valid to perform a 0 Auth on a customer's account. 0 Auth means it does not hold any funds on the customer’s account.

Note: The card information is tokenized into an encrypted string and tested as part of the pre-authorisation process. Not all gateways support this feature.

Warning: The Issuing Bank will not check funds.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

API Key (header) Api-Token

For Payment Session authentication, supply the token used to authenticate the call to generate a payment session

PaymentSessionAuthReference section must also be completed.

Enter api key (header) value

API Key (header) Payment-Session

For Payment Session authentication, supply the reference returned in the create payment session response

PaymentSessionAuthToken section must also be completed.

Enter api key (header) value

Header Parameters

> **Api-Version** `string` *

Body Parameters

body `checkCardRequestPan` *

All Of
cardDetailsCv2N...

Description

Request is sent with full card details

- **cardNumber** `string` *
- **cv2** `string`
- **expiryDate** `string` *
- **startDate** `string`
- **yourConsumerReference** `string` *
- **yourPaymentReference** `string`
- **clientDetails** `object`
- **judoId** `string`
- **currency** `string`
- **initialRecurringPayment** `boolean`
- **disableNetworkTokenisation** `boolean`
- **cardHolderName** `string`
- **emailAddress** `string`
- **shippingAddress** `object`
- **mobileNumber** `string`
- **phoneCountryCode** `string` — 44
- **threeDSecure** `object`
- **threeDSecureMpi** `object`
- **cardAddress** `cardAddressRequestAttributes` — All Of: cardAddressCom...

> **primaryAccountDetails** `object`

Responses

200 Successful response will be one of
* transaction receipt
* request for 3DS2 device details
* request for 3DS2 challenge

object

One Of
transactionRecei...

- receiptId string
- originalReceiptId string
- yourPaymentReference string
- type string<Payment | Refund | PreAuth…
- createdAt string<date-time>
- result string
- message string
- judoId integer
- merchantName string
- appearsOnStatementAs string
- originalAmount string
- amountCollected string
- netAmount string
- amount string
- currency string
- recurringPaymentType string
- acquirerTransactionId string
- externalBankResponseCode string
- authCode string
- walletType integer
- riskScore integer

- paymentNetworkTransactionId `string`
- allowIncrement `boolean`
- isIncrementalAuth `boolean`
- emailAddress `string`
- cardDetails `cardDetailsResponse`
- billingAddress `cardAddressResponseAttribu…` All Of **cardAddressCom…**
- consumer `consumerResponse`
- device `deviceResponse`
- yourPaymentMetaData `object`
- threeDSecure `threeDSecureCompletedRespo…`
- risks `riskParametersResponse`
- disableNetworkTokenisation `boolean`
- networkTokenisationDetails `networkTokenisa…`

● **400** Bad request if any of the required request parameters are not supplied

● **401** Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

● **403** Forbidden if token used does not have required permission

- errorResponse
  - requestId string
  - message string
  - code integer
  - category integer

```
curl --request POST \
     --url https://api-sandbox.judopay.com/transactions/checkcard \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "cardNumber": 1234567890123456,
     "expiryDate": "01/28",
     "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
     "phoneCountryCode": "44",
     "threeDSecure": {
     "methodNotificationUrl": "https://api.judopay.com/order/3ds/methodNotification",
     "challengeNotificationUrl": "https://api.judopay.com/order/3ds/challengeNotificatio
     },
     "threeDSecureMpi": {
     "dsTransId": "41aadf4c-e73f-4bd1-a0c4-acb2615a32af",
     "cavv": "AJkBCWUyZwAAAABugmKTdAAAAAA=",
     "eci": "02",
     "threeDSecureVersion": "2.2.0"
     },
     "cardAddress": {
     "postCode": "AB1 2CD"
     },
     "primaryAccountDetails": {
     "name": "Doe",
     "accountNumber": "12345678",
     "dateOfBirth": "1980-01-31",
     "postCode": "AB1 2CD"
     }
     }'
```

```
// Successful response will be one of
  * transaction receipt
  * request for 3DS2 device details
  * request for 3DS2 challenge
{}
```

24.2.2. /transactions/registercard

POST https://api-sandbox.judopay.com/transactions/registercard Try it

Sandbox environment

Warning: Deprecated
Conduct a pre-authorisation, reserving a pre-configured amount in the customer's account. The amount is set during onboarding (default is 1.01). Registercard will tokenise the card information into an encrypted string.
The POST /transactions/voids endpoint can be used to cancel this pre-authorisation so it does not appear in the customer's statement

Note: The card information is tokenised into an encrypted string and tested as part of the pre-authorisation process.

This call has been deprecated in favour of POST /transactions/checkcard or POST /transactions/preauths

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

API Key (header) Api-Token

For Payment Session authentication, supply the token used to authenticate the call to generate a payment session

PaymentSessionAuthReference section must also be completed.

Enter api key (header) value

API Key (header) Payment-Session

For Payment Session authentication, supply the reference returned in the create payment session response

PaymentSessionAuthToken section must also be completed.

Enter api key (header) value

Header Parameters

> **Api-Version** `string` *

Body Parameters

body `registerCardRequestPan` *

All Of
cardDetailsCv2N...

Description

Request is sent with full card details

- **cardNumber** `string` *
- **cv2** `string`
- **expiryDate** `string` *
- **startDate** `string`
- **yourConsumerReference** `string` *
- **yourPaymentReference** `string`
- **clientDetails** `object`
- **judoId** `string`
- **currency** `string`
- **initialRecurringPayment** `boolean`
- **cardHolderName** `string`
- **emailAddress** `string`
- **shippingAddress** `object`
- **mobileNumber** `string`
- **phoneCountryCode** `string` 44
- **threeDSecure** `object`
- **threeDSecureMpi** `object`
- **cardAddress** `cardAddressRequestAttributes`

All Of
cardAddressCom...

- **primaryAccountDetails** `object`

Responses

200 Successful response will be one of
* transaction receipt
* request for 3DS2 device details
* request for 3DS2 challenge

object

One Of
transactionRecei...

- receiptId string
- originalReceiptId string
- yourPaymentReference string
- type string<Payment | Refund | PreAuth…
- createdAt string<date-time>
- result string
- message string
- judoId integer
- merchantName string
- appearsOnStatementAs string
- originalAmount string
- amountCollected string
- netAmount string
- amount string
- currency string
- recurringPaymentType string
- acquirerTransactionId string
- externalBankResponseCode string
- authCode string
- walletType integer
- riskScore integer

- paymentNetworkTransactionId `string`
- allowIncrement `boolean`
- isIncrementalAuth `boolean`
- emailAddress `string`
- cardDetails `cardDetailsResponse`
- billingAddress `cardAddressResponseAttribu…` All Of cardAddressCom…
- consumer `consumerResponse`
- device `deviceResponse`
- yourPaymentMetaData `object`
- threeDSecure `threeDSecureCompletedRespo…`
- risks `riskParametersResponse`
- disableNetworkTokenisation `boolean`
- networkTokenisationDetails `networkTokenisa…`

● 400 Bad request if any of the required request parameters are not supplied

● 401 Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

● 403 Forbidden if token used does not have required permission

- errorResponse
  - requestId string
  - message string
  - code integer
  - category integer

```
curl --request POST \
     --url https://api-sandbox.judopay.com/transactions/registercard \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "cardNumber": 1234567890123456,
     "expiryDate": "01/28",
     "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
     "phoneCountryCode": "44",
     "threeDSecure": {
     "methodNotificationUrl": "https://api.judopay.com/order/3ds/methodNotification",
     "challengeNotificationUrl": "https://api.judopay.com/order/3ds/challengeNotificatio
     "challengeRequestIndicator": "noPreference"
     },
     "threeDSecureMpi": {
     "dsTransId": "41aadf4c-e73f-4bd1-a0c4-acb2615a32af",
     "cavv": "AJkBCWUyZwAAAABugmKTdAAAAAA=",
     "eci": "02",
     "threeDSecureVersion": "2.2.0"
     },
     "cardAddress": {
     "postCode": "AB1 2CD"
     },
     "primaryAccountDetails": {
     "name": "Doe",
     "accountNumber": "12345678",
     "dateOfBirth": "1980-01-31",
     "postCode": "AB1 2CD"
     }
     }'
```

```
// Successful response will be one of
  * transaction receipt
  * request for 3DS2 device details
  * request for 3DS2 challenge
{}
```

24.2.3. /transactions/preauths

POST https://api-sandbox.judopay.com/transactions/preauths Try it

Sandbox environment

Use preauths to reserve funds on a customer's account and postpone completing the payment until the goods have been delivered or the service fulfilled. This is also known as reserve and collection of funds.
Caution The expiry time of the reserved pre-authorised funds differs across issuers.

Note To reduce your PCI scope, you can store a card token instead of the card number. The card token is a randomly generated string linked to a card saved securely within the Judopay Card Vault. You can also use a card token to pre-authorise repeat payments without the consumer having to re-enter their details.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

API Key (header) Api-Token

For Payment Session authentication, supply the token used to authenticate the call to generate a payment session

PaymentSessionAuthReference section must also be completed.

Enter api key (header) value

API Key (header) Payment-Session

For Payment Session authentication, supply the reference returned in the create payment session response

PaymentSessionAuthToken section must also be completed.

Enter api key (header) value

Header Parameters

Api-Version `string` *

Body Parameters

body `object` *

Description

Request can be sent with
  * full card details,
  * a cardToken generated with a call to [POST /transactions/savecard](), or
  * wallet details from Apple Pay or Google Pay

One Of
preAuthRequestP...

- **cardNumber** `string` *
- **cv2** `string`
- **expiryDate** `string` *
- **startDate** `string`
- **cardAddress** `cardAddressRequestAttributes`

All Of
cardAddressCom...

- **judoId** `string` *
- **yourConsumerReference** `string` *
- **yourPaymentReference** `string` *
- **yourPaymentMetaData** `object` *
- **currency** `string` *
- **amount** `number<float>` *
- **webPaymentReference** `string` *
- **clientDetails** `object` *
- **initialRecurringPayment** `boolean` *
- **primaryAccountDetails** `object` *
- **acceptHeaders** `string` text/html
- **userAgent** `string`

- **cardHolderName** `string`
- **emailAddress** `string`
- **shippingAddress** `object`
- **mobileNumber** `string`
- **phoneCountryCode** `string` — 44
- **threeDSecure** `object`
- **threeDSecureMpi** `object`
- **recurringPayment** `boolean`
- **recurringPaymentType** `string<RECURRING | MI…` — Select...
- **relatedReceiptId** `string`
- **relatedPaymentNetworkTransactionId** `string`
- **allowIncrement** `boolean`
- **disableNetworkTokenisation** `boolean`

Responses

- **200** Successful response will be one of
  * transaction receipt
  * request for 3DS2 device details
  * request for 3DS2 challenge

object

One Of
transactionRecei...

- receiptId string
- originalReceiptId string
- yourPaymentReference string
- type string<Payment | Refund | PreAuth…
- createdAt string<date-time>
- result string
- message string
- judoId integer
- merchantName string
- appearsOnStatementAs string
- originalAmount string
- amountCollected string
- netAmount string
- amount string
- currency string
- recurringPaymentType string
- acquirerTransactionId string
- externalBankResponseCode string
- authCode string
- walletType integer
- riskScore integer

- **paymentNetworkTransactionId** `string`
- **allowIncrement** `boolean`
- **isIncrementalAuth** `boolean`
- **emailAddress** `string`
- **cardDetails** `cardDetailsResponse`
- **billingAddress** `cardAddressResponseAttribu…` All Of `cardAddressCom…`
- **consumer** `consumerResponse`
- **device** `deviceResponse`
- **yourPaymentMetaData** `object`
- **threeDSecure** `threeDSecureCompletedRespo…`
- **risks** `riskParametersResponse`
- **disableNetworkTokenisation** `boolean`
- **networkTokenisationDetails** `networkTokenisa…`

● **400** Bad request if any of the required request parameters are not supplied

● **401** Unauthorized

`errorResponse`

- **requestId** `string`
- **message** `string`
- **code** `integer`
- **category** `integer`

● **403** Forbidden if token used does not have required permission

errorResponse

- requestId string
- message string
- code integer
- category integer

409 Conflict - unable to find a route for the transactions

errorResponse

- requestId string
- message string
- code integer
- category integer

```
curl --request POST \
     --url https://api-sandbox.judopay.com/transactions/preauths \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "cardNumber": 1234567890123456,
     "expiryDate": "01/28",
     "cardAddress": {
     "postCode": "AB1 2CD"
     },
     "judoId": "100100100",
     "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
     "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
     "yourPaymentMetaData": "{\"internalLocationRef\":\"Example\",\"internalId\":99}",
     "currency": "GBP",
     "amount": 10.99,
     "webPaymentReference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7J
     "clientDetails": "",
     "initialRecurringPayment": false,
     "primaryAccountDetails": "",
     "acceptHeaders": "text/html",
     "phoneCountryCode": "44",
     "threeDSecure": {
     "methodNotificationUrl": "https://api.judopay.com/order/3ds/methodNotification",
     "challengeNotificationUrl": "https://api.judopay.com/order/3ds/challengeNotificatio
     "challengeRequestIndicator": "noPreference"
     },
     "threeDSecureMpi": {
     "dsTransId": "41aadf4c-e73f-4bd1-a0c4-acb2615a32af",
     "cavv": "AJkBCWUyZwAAAABugmKTdAAAAAA=",
     "eci": "02",
     "threeDSecureVersion": "2.2.0"
     }
     }'
```

```
// Successful response will be one of
  * transaction receipt
  * request for 3DS2 device details
  * request for 3DS2 challenge
{}
```

24.2.4. /transactions/incrementalAuth

POST https://api-sandbox.judopay.com/transactions/incrementalAuth Try it

Sandbox environment

Increment the amount on a successful customer initiated pre-auth (before it is fully collected).

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

- **Api-Version** `string` *

Body Parameters

- **body** `incrementAuthRequest` *
  - **receiptId** `string` *
  - **amount** `number<float>` *
  - **yourPaymentMetaData** `object`
  - **yourPaymentReference** `string` *

Responses

- 200 Successful response

- transactionReceiptResponse
  - receiptId string
  - originalReceiptId string
  - yourPaymentReference string
  - type string<Payment | Refund | PreAuth…
  - createdAt string<date-time>
  - result string
  - message string
  - judoId integer
  - merchantName string
  - appearsOnStatementAs string
  - originalAmount string
  - amountCollected string
  - netAmount string
  - amount string
  - currency string
  - recurringPaymentType string
  - acquirerTransactionId string
  - externalBankResponseCode string
  - authCode string
  - walletType integer
  - riskScore integer

- paymentNetworkTransactionId `string`
- allowIncrement `boolean`
- isIncrementalAuth `boolean`
- emailAddress `string`
- cardDetails `cardDetailsResponse`
- billingAddress `cardAddressResponseAttribu…` All Of cardAddressCom…
- consumer `consumerResponse`
- device `deviceResponse`
- yourPaymentMetaData `object`
- threeDSecure `threeDSecureCompletedRespo…`
- risks `riskParametersResponse`
- disableNetworkTokenisation `boolean`
- networkTokenisationDetails `networkTokenisa…`

● 401 Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

● 403 Forbidden if token used does not have required permission

errorResponse

- requestId string
- message string
- code integer
- category integer

```
curl --request POST \
     --url https://api-sandbox.judopay.com/transactions/incrementalAuth \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "receiptId": "1001131610340495360",
     "amount": 0.99,
     "yourPaymentReference": "aa648425-2230-4e4e-8a61-27f06df54542"
     }'
```

```
// Successful response
{
  "receiptId": "1001131610340495360",
  "originalReceiptId": "1001124307998347264",
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "type": "PreAuth",
  "createdAt": "2025-02-05T16:28:32.8596+00:00",
  "result": "Success",
  "message": "AuthCode: 123456",
  "judoId": 100100100,
  "merchantName": "Test Merchant",
  "appearsOnStatementAs": "APL*/TestMerchant       ",
  "originalAmount": "10.99",
  "amountCollected": "1.99",
  "netAmount": "1.00",
  "amount": "1.00",
  "currency": "GBP",
  "recurringPaymentType": "MIT",
  "acquirerTransactionId": "33666277256892153705",
  "externalBankResponseCode": "A",
  "authCode": "123456",
  "walletType": 1,
  "riskScore": 0,
  "paymentNetworkTransactionId": "123456789012345",
  "allowIncrement": true,
  "isIncrementalAuth": true,
```

24.2.5. /transactions/collections

POST https://api-sandbox.judopay.com/transactions/collections Try it

Sandbox environment

Collect funds previously reserved using the pre-authorisation request. You can choose to let your customer know when to expect their card to be charged:

You have not yet been charged for this item. We won't debit any funds from your payment card until your journey is complete.

Caution The pre-authorised funds will expire in three days if not collected, releasing the reserved funds.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

> Api-Version `string` *

Body Parameters

body `collectionRequest` *

- **receiptId** `string` *
- **amount** `number<float>`
- **yourPaymentMetaData** `object`
- **yourPaymentReference** `string` *

Responses

200 Successful response

- transactionReceiptResponse
  - receiptId string
  - originalReceiptId string
  - yourPaymentReference string
  - type string<Payment | Refund | PreAuth…
  - createdAt string<date-time>
  - result string
  - message string
  - judoId integer
  - merchantName string
  - appearsOnStatementAs string
  - originalAmount string
  - amountCollected string
  - netAmount string
  - amount string
  - currency string
  - recurringPaymentType string
  - acquirerTransactionId string
  - externalBankResponseCode string
  - authCode string
  - walletType integer
  - riskScore integer

- paymentNetworkTransactionId `string`
- allowIncrement `boolean`
- isIncrementalAuth `boolean`
- emailAddress `string`
- cardDetails `cardDetailsResponse`
- billingAddress `cardAddressResponseAttribu…` All Of cardAddressCom…
- consumer `consumerResponse`
- device `deviceResponse`
- yourPaymentMetaData `object`
- threeDSecure `threeDSecureCompletedRespo…`
- risks `riskParametersResponse`
- disableNetworkTokenisation `boolean`
- networkTokenisationDetails `networkTokenisa…`

● 401 Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

● 403 Forbidden if token used does not have required permission

- errorResponse
  - requestId string
  - message string
  - code integer
  - category integer

```
curl --request POST \
     --url https://api-sandbox.judopay.com/transactions/collections \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "receiptId": "1001131610340495360",
     "yourPaymentReference": "aa648425-2230-4e4e-8a61-27f06df54542"
     }'
```

```
// Successful response
{
  "receiptId": "1001131610340495360",
  "originalReceiptId": "1001124307998347264",
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "type": "PreAuth",
  "createdAt": "2025-02-05T16:28:32.8596+00:00",
  "result": "Success",
  "message": "AuthCode: 123456",
  "judoId": 100100100,
  "merchantName": "Test Merchant",
  "appearsOnStatementAs": "APL*/TestMerchant       ",
  "originalAmount": "10.99",
  "amountCollected": "1.99",
  "netAmount": "1.00",
  "amount": "1.00",
  "currency": "GBP",
  "recurringPaymentType": "MIT",
  "acquirerTransactionId": "33666277256892153705",
  "externalBankResponseCode": "A",
  "authCode": "123456",
  "walletType": 1,
  "riskScore": 0,
  "paymentNetworkTransactionId": "123456789012345",
  "allowIncrement": true,
  "isIncrementalAuth": true,
```

24.2.6. /transactions/voids

POST https://api-sandbox.judopay.com/transactions/voids Try it

Sandbox environment

Use void to cancel a pre-authorised transaction if the funds have not yet settled. Note Not all banks support voiding transactions. Check with your Acquirer. Voids cannot be performed on transactions that have been collected (partial or full).

Caution The pre-authorised funds will expire in three days if not collected, releasing the reserved funds.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

> Api-Version `string` *

Body Parameters

body `voidRequest` *

- **receiptId** `string` *
- **amount** `number<float>`
- **yourPaymentReference** `string`

Responses

- **200** Successful response

- transactionReceiptResponse
  - receiptId string
  - originalReceiptId string
  - yourPaymentReference string
  - type string<Payment | Refund | PreAuth…
  - createdAt string<date-time>
  - result string
  - message string
  - judoId integer
  - merchantName string
  - appearsOnStatementAs string
  - originalAmount string
  - amountCollected string
  - netAmount string
  - amount string
  - currency string
  - recurringPaymentType string
  - acquirerTransactionId string
  - externalBankResponseCode string
  - authCode string
  - walletType integer
  - riskScore integer

- paymentNetworkTransactionId `string`
- allowIncrement `boolean`
- isIncrementalAuth `boolean`
- emailAddress `string`
- cardDetails `cardDetailsResponse`
- billingAddress `cardAddressResponseAttribu…` All Of cardAddressCom…
- consumer `consumerResponse`
- device `deviceResponse`
- yourPaymentMetaData `object`
- threeDSecure `threeDSecureCompletedRespo…`
- risks `riskParametersResponse`
- disableNetworkTokenisation `boolean`
- networkTokenisationDetails `networkTokenisa…`

● 401 Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

● 403 Forbidden if token used does not have required permission

- errorResponse
  - requestId string
  - message string
  - code integer
  - category integer

```
curl --request POST \
     --url https://api-sandbox.judopay.com/transactions/voids \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "receiptId": "1001131610340495360"
     }'
```

```
// Successful response
{
  "receiptId": "1001131610340495360",
  "originalReceiptId": "1001124307998347264",
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "type": "PreAuth",
  "createdAt": "2025-02-05T16:28:32.8596+00:00",
  "result": "Success",
  "message": "AuthCode: 123456",
  "judoId": 100100100,
  "merchantName": "Test Merchant",
  "appearsOnStatementAs": "APL*/TestMerchant       ",
  "originalAmount": "10.99",
  "amountCollected": "1.99",
  "netAmount": "1.00",
  "amount": "1.00",
  "currency": "GBP",
  "recurringPaymentType": "MIT",
  "acquirerTransactionId": "33666277256892153705",
  "externalBankResponseCode": "A",
  "authCode": "123456",
  "walletType": 1,
  "riskScore": 0,
  "paymentNetworkTransactionId": "123456789012345",
  "allowIncrement": true,
  "isIncrementalAuth": true,
```

24.3. Payments

24.3.1. /transactions/payments

POST https://api-sandbox.judopay.com/transactions/payments Try it

Sandbox environment

Note To reduce your PCI scope, you can store a card token instead of the card number. The card token is a randomly generated string linked to a card saved securely within the Judopay Card Vault. You can also use a card token to pre-authorise repeat payments without the consumer having to re-enter their details.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

API Key (header) Api-Token

For Payment Session authentication, supply the token used to authenticate the call to generate a payment session

PaymentSessionAuthReference section must also be completed.

Enter api key (header) value

API Key (header) Payment-Session

For Payment Session authentication, supply the reference returned in the create payment session response

PaymentSessionAuthToken section must also be completed.

Enter api key (header) value

Header Parameters

> **Api-Version** `string` *

Body Parameters

body `object` *

Description

Request can be sent with
  * full card details,
  * a cardToken generated with a call to [POST /transactions/savecard](), or
  * wallet details from Apple Pay or Google Pay

One Of
paymentRequest...

- **cardNumber** `string` *
- **cv2** `string`
- **expiryDate** `string` *
- **startDate** `string`
- **cardAddress** `cardAddressRequestAttributes`

All Of
cardAddressCom...

- **judoId** `string` *
- **yourConsumerReference** `string` *
- **yourPaymentReference** `string` *
- **yourPaymentMetaData** `object` *
- **currency** `string` *
- **amount** `number<float>` *
- **webPaymentReference** `string` *
- **clientDetails** `object` *
- **initialRecurringPayment** `boolean` *
- **primaryAccountDetails** `object` *
- **acceptHeaders** `string` text/html
- **userAgent** `string`

- **cardHolderName** `string`
- **emailAddress** `string`
- **shippingAddress** `object`
- **mobileNumber** `string`
- **phoneCountryCode** `string` — 44
- **threeDSecure** `object`
- **threeDSecureMpi** `object`
- **recurringPayment** `boolean`
- **recurringPaymentType** `string<RECURRING | MI…` — Select...
- **relatedReceiptId** `string`
- **relatedPaymentNetworkTransactionId** `string`
- **disableNetworkTokenisation** `boolean`

Responses

- **200** Successful response will be one of
  * transaction receipt
  * request for 3DS2 device details
  * request for 3DS2 challenge

object

One Of
transactionRecei...

- receiptId string
- originalReceiptId string
- yourPaymentReference string
- type string<Payment | Refund | PreAuth…
- createdAt string<date-time>
- result string
- message string
- judoId integer
- merchantName string
- appearsOnStatementAs string
- originalAmount string
- amountCollected string
- netAmount string
- amount string
- currency string
- recurringPaymentType string
- acquirerTransactionId string
- externalBankResponseCode string
- authCode string
- walletType integer
- riskScore integer

- paymentNetworkTransactionId `string`
- allowIncrement `boolean`
- isIncrementalAuth `boolean`
- emailAddress `string`
- cardDetails `cardDetailsResponse`
- billingAddress `cardAddressResponseAttribu…` All Of cardAddressCom…
- consumer `consumerResponse`
- device `deviceResponse`
- yourPaymentMetaData `object`
- threeDSecure `threeDSecureCompletedRespo…`
- risks `riskParametersResponse`
- disableNetworkTokenisation `boolean`
- networkTokenisationDetails `networkTokenisa…`

● 401 Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

● 403 Forbidden if token used does not have required permission

errorResponse

- **requestId** `string`
- **message** `string`
- **code** `integer`
- **category** `integer`

409 Conflict - unable to find a route for the transactions

errorResponse

- **requestId** `string`
- **message** `string`
- **code** `integer`
- **category** `integer`

```
curl --request POST \
     --url https://api-sandbox.judopay.com/transactions/payments \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "cardNumber": 1234567890123456,
     "expiryDate": "01/28",
     "cardAddress": {
     "postCode": "AB1 2CD"
     },
     "judoId": "100100100",
     "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
     "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
     "yourPaymentMetaData": "{\"internalLocationRef\":\"Example\",\"internalId\":99}",
     "currency": "GBP",
     "amount": 10.99,
     "webPaymentReference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7JX
     "clientDetails": "",
     "initialRecurringPayment": false,
     "primaryAccountDetails": "",
     "acceptHeaders": "text/html",
     "phoneCountryCode": "44",
     "threeDSecure": {
     "methodNotificationUrl": "https://api.judopay.com/order/3ds/methodNotification",
     "challengeNotificationUrl": "https://api.judopay.com/order/3ds/challengeNotificatio
     "challengeRequestIndicator": "noPreference"
     },
     "threeDSecureMpi": {
     "dsTransId": "41aadf4c-e73f-4bd1-a0c4-acb2615a32af",
     "cavv": "AJkBCWUyZwAAAABugmKTdAAAAAA=",
     "eci": "02",
     "threeDSecureVersion": "2.2.0"
     }
     }'
```

```
// Successful response will be one of
  * transaction receipt
  * request for 3DS2 device details
  * request for 3DS2 challenge
{}
```

24.3.2. /transactions/refunds

POST https://api-sandbox.judopay.com/transactions/refunds Try it

Sandbox environment

Use refund to return funds to the customer. You can process a full or partial refund:

- Full Refund. Returns the total of the original transaction back to the customer.
- Partial Refund. Returns a specified amount of the original transaction back to the customer. For example, if a customer has been overcharged, or where part of an order can't be satisfied by the business.

** Note ** Not all Acquiring banks support partial refunds on the same date as the original transaction.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

> Api-Version `string` *

Body Parameters

body `refundRequest` *

> receiptId `string` *

> amount `number<float>`

> yourPaymentReference `string` *

Responses

200 Successful response

- transactionReceiptResponse
  - receiptId string
  - originalReceiptId string
  - yourPaymentReference string
  - type string<Payment | Refund | PreAuth…
  - createdAt string<date-time>
  - result string
  - message string
  - judoId integer
  - merchantName string
  - appearsOnStatementAs string
  - originalAmount string
  - amountCollected string
  - netAmount string
  - amount string
  - currency string
  - recurringPaymentType string
  - acquirerTransactionId string
  - externalBankResponseCode string
  - authCode string
  - walletType integer
  - riskScore integer

- paymentNetworkTransactionId `string`
- allowIncrement `boolean`
- isIncrementalAuth `boolean`
- emailAddress `string`
- cardDetails `cardDetailsResponse`
- billingAddress `cardAddressResponseAttribu…` All Of cardAddressCom…
- consumer `consumerResponse`
- device `deviceResponse`
- yourPaymentMetaData `object`
- threeDSecure `threeDSecureCompletedRespo…`
- risks `riskParametersResponse`
- disableNetworkTokenisation `boolean`
- networkTokenisationDetails `networkTokenisa…`

● 401 Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

● 403 Forbidden if token used does not have required permission

- errorResponse
  - requestId string
  - message string
  - code integer
  - category integer

```
curl --request POST \
     --url https://api-sandbox.judopay.com/transactions/refunds \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "receiptId": "1001131610340495360",
     "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134"
     }'
```

```
// Successful response
{
  "receiptId": "1001131610340495360",
  "originalReceiptId": "1001124307998347264",
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "type": "PreAuth",
  "createdAt": "2025-02-05T16:28:32.8596+00:00",
  "result": "Success",
  "message": "AuthCode: 123456",
  "judoId": 100100100,
  "merchantName": "Test Merchant",
  "appearsOnStatementAs": "APL*/TestMerchant       ",
  "originalAmount": "10.99",
  "amountCollected": "1.99",
  "netAmount": "1.00",
  "amount": "1.00",
  "currency": "GBP",
  "recurringPaymentType": "MIT",
  "acquirerTransactionId": "33666277256892153705",
  "externalBankResponseCode": "A",
  "authCode": "123456",
  "walletType": 1,
  "riskScore": 0,
  "paymentNetworkTransactionId": "123456789012345",
  "allowIncrement": true,
  "isIncrementalAuth": true,
```

24.4. 3DSecure

24.4.1. /transactions/{receiptId}/resume3ds

PUT https://api-sandbox.judopay.com/transactions/{receiptId}/resume3d Try it

Sandbox environment

Use /resume3ds when authenticating with 3D Secure 2, following the 3D Secure 2 device details being successfully completed by the consumer.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

API Key (header) Api-Token

For Payment Session authentication, supply the token used to authenticate the call to generate a payment session

PaymentSessionAuthReference section must also be completed.

Enter api key (header) value

API Key (header) Payment-Session

For Payment Session authentication, supply the reference returned in the create payment session response

PaymentSessionAuthToken section must also be completed.

Enter api key (header) value

Path Parameters

> **receiptId** `string` *

Header Parameters

> **Api-Version** `string` *

Body Parameters

> **body** `resume3dsRequest` *

All Of
object +

> **cv2** `string`

> **threeDSecure** `object` *

> **primaryAccountDetails** `object`

Responses

> ● **200** Successful response

object

One Of
transactionRecei...

- receiptId string
- originalReceiptId string
- yourPaymentReference string
- type string<Payment | Refund | PreAuth…
- createdAt string<date-time>
- result string
- message string
- judoId integer
- merchantName string
- appearsOnStatementAs string
- originalAmount string
- amountCollected string
- netAmount string
- amount string
- currency string
- recurringPaymentType string
- acquirerTransactionId string
- externalBankResponseCode string
- authCode string
- walletType integer
- riskScore integer

- paymentNetworkTransactionId `string`
- allowIncrement `boolean`
- isIncrementalAuth `boolean`
- emailAddress `string`
- cardDetails `cardDetailsResponse`
- billingAddress `cardAddressResponseAttribu…` All Of **cardAddressCom…**
- consumer `consumerResponse`
- device `deviceResponse`
- yourPaymentMetaData `object`
- threeDSecure `threeDSecureCompletedRespo…`
- risks `riskParametersResponse`
- disableNetworkTokenisation `boolean`
- networkTokenisationDetails `networkTokenisa…`

400 Bad request (Api-Version header missing, or request body attributes invalid)

401 Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

```
curl --request PUT \
     --url https://api-sandbox.judopay.com/transactions/{receiptId}/resume3ds \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "threeDSecure": {
     "methodCompletion": "Yes"
     },
     "primaryAccountDetails": {
     "name": "Doe",
     "accountNumber": "12345678",
     "dateOfBirth": "1980-01-31",
     "postCode": "AB1 2CD"
     }
     }'
```

```
// Successful response
{}
```

24.4.2. /transactions/{receiptId}/complete3ds

PUT https://api-sandbox.judopay.com/transactions/{receiptId}/complete3 Try it

Sandbox environment

Use this endpoint when authenticating with 3D Secure 2, following the 3D Secure 2 challenge being successfully completed by the consumer.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

API Key (header) Api-Token

For Payment Session authentication, supply the token used to authenticate the call to generate a payment session

PaymentSessionAuthReference section must also be completed.

Enter api key (header) value

API Key (header) Payment-Session

For Payment Session authentication, supply the reference returned in the create payment session response

PaymentSessionAuthToken section must also be completed.

Enter api key (header) value

Path Parameters

receiptId `string` *

Header Parameters

Api-Version `string` *

Body Parameters

body `complete3ds2Request` *

All Of
object +

- **cv2** `string`
- **version** `string` *
- **primaryAccountDetails** `object`

Responses

- **200** Successful response

- transactionReceiptResponse
  - receiptId string
  - originalReceiptId string
  - yourPaymentReference string
  - type string<Payment | Refund | PreAuth…
  - createdAt string<date-time>
  - result string
  - message string
  - judoId integer
  - merchantName string
  - appearsOnStatementAs string
  - originalAmount string
  - amountCollected string
  - netAmount string
  - amount string
  - currency string
  - recurringPaymentType string
  - acquirerTransactionId string
  - externalBankResponseCode string
  - authCode string
  - walletType integer
  - riskScore integer

- **paymentNetworkTransactionId** `string`
- **allowIncrement** `boolean`
- **isIncrementalAuth** `boolean`
- **emailAddress** `string`
- **cardDetails** `cardDetailsResponse`
- **billingAddress** `cardAddressResponseAttribu…` All Of `cardAddressCom…`
- **consumer** `consumerResponse`
- **device** `deviceResponse`
- **yourPaymentMetaData** `object`
- **threeDSecure** `threeDSecureCompletedRespo…`
- **risks** `riskParametersResponse`
- **disableNetworkTokenisation** `boolean`
- **networkTokenisationDetails** `networkTokenisa…`

● **400** Bad request (Api-Version header missing, or request body attributes invalid)

● **401** Unauthorized

`errorResponse`

- **requestId** `string`
- **message** `string`
- **code** `integer`
- **category** `integer`

```
curl --request PUT \
     --url https://api-sandbox.judopay.com/transactions/{receiptId}/complete3ds \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "version": "2.2.0",
     "primaryAccountDetails": {
     "name": "Doe",
     "accountNumber": "12345678",
     "dateOfBirth": "1980-01-31",
     "postCode": "AB1 2CD"
     }
     }'
```

```
// Successful response
{
  "receiptId": "1001131610340495360",
  "originalReceiptId": "1001124307998347264",
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "type": "PreAuth",
  "createdAt": "2025-02-05T16:28:32.8596+00:00",
  "result": "Success",
  "message": "AuthCode: 123456",
  "judoId": 100100100,
  "merchantName": "Test Merchant",
  "appearsOnStatementAs": "APL*/TestMerchant       ",
  "originalAmount": "10.99",
  "amountCollected": "1.99",
  "netAmount": "1.00",
  "amount": "1.00",
  "currency": "GBP",
  "recurringPaymentType": "MIT",
  "acquirerTransactionId": "33666277256892153705",
  "externalBankResponseCode": "A",
  "authCode": "123456",
  "walletType": 1,
  "riskScore": 0,
  "paymentNetworkTransactionId": "123456789012345",
  "allowIncrement": true,
  "isIncrementalAuth": true,
```

24.5. Receipts

24.5.1. /transactions/{receiptId}

GET https://api-sandbox.judopay.com/transactions/{receiptId} Try it

Sandbox environment

Retrieve details of a single transaction. Get the receiptId from the response of the initial payment/preauth request.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Path Parameters

receiptId `string` *

Header Parameters

Api-Version `string` *

Responses

200 Successful response

- transactionReceiptHistoricResponse
  - receiptId string
  - originalReceiptId string
  - yourPaymentReference string
  - type string<Payment | Refund | PreAuth…
  - createdAt string<date-time>
  - result string
  - message string
  - judoId integer
  - merchantName string
  - appearsOnStatementAs string
  - originalAmount string
  - amountCollected string
  - netAmount string
  - amount string
  - currency string
  - recurringPaymentType string
  - acquirerTransactionId string
  - externalBankResponseCode string
  - authCode string
  - postCodeCheckResult string
  - walletType integer

- acquirer `string`
- webPaymentReference `string`
- noOfAuthAttempts `integer`
- paymentNetworkTransactionId `string`
- allowIncrement `boolean`
- isIncrementalAuth `boolean`
- disableNetworkTokenisation `boolean`
- cardDetails `cardDetailsResponse`
- billingAddress `cardAddressResponseAttribu…` All Of cardAddressCom…
- consumer `consumerResponse`
- device `deviceResponse`
- yourPaymentMetaData `object`
- threeDSecure `threeDSecureCompletedRespo…`
- risks `riskParametersResponse`
- networkTokenisationDetails `networkTokenisa…`

● 401 Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

● 403 Forbidden if token used does not have required permission

errorResponse

- requestId string
- message string
- code integer
- category integer

404 Not Found

errorResponse

- requestId string
- message string
- code integer
- category integer

```
curl --request GET \
     --url https://api-sandbox.judopay.com/transactions/{receiptId} \
     --header 'accept: application/json' \
     --header 'api-version: 6.23'
```

```
// Successful response
{
  "receiptId": "1001131610340495360",
  "originalReceiptId": "1001124307998347264",
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "type": "PreAuth",
  "createdAt": "2025-02-05T16:28:32.8596+00:00",
  "result": "Success",
  "message": "AuthCode: 123456",
  "judoId": 100100100,
  "merchantName": "Test Merchant",
  "appearsOnStatementAs": "APL*/TestMerchant       ",
  "originalAmount": "10.99",
  "amountCollected": "1.99",
  "netAmount": "1.00",
  "amount": "1.00",
  "currency": "GBP",
  "recurringPaymentType": "MIT",
  "acquirerTransactionId": "33666277256892153705",
  "externalBankResponseCode": "A",
  "authCode": "123456",
  "postCodeCheckResult": "Passed",
  "walletType": 1,
  "acquirer": "MyBank",
  "webPaymentReference": "5QcAAAMAAAASAAAADAAAAHaKm0p0p-Ew6VrBhPxjxvMLinJOMZEJWG_K7kJ53U
  "noOfAuthAttempts": 1,
  "paymentNetworkTransactionId": "123456789012345",
  "allowIncrement": true,
```

24.5.2. /transactions

GET https://api-sandbox.judopay.com/transactions Try it

Sandbox environment

Return a list of transactions associated with the given query. Please note: The records returned by this endpoint should not be relied upon for real-time processing.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

- **Api-Version** `string` *

Query Parameters

- **pageSize** `integer` — 10
- **offset** `integer`
- **sort** `string` — time-descending
- **from** `string`
- **to** `string`
- **yourPaymentReference** `string`
- **yourConsumerReference** `string`

Responses

- 200 Successful response

listTransactionsResponse

- **resultCount** number
- **pageSize** integer
- **offset** integer
- **results** transactionReceiptListResponse[]
- **sort** string

401 Unauthorized

errorResponse

- **requestId** string
- **message** string
- **code** integer
- **category** integer

403 Forbidden if token used does not have required permission

errorResponse

- **requestId** string
- **message** string
- **code** integer
- **category** integer

```
curl --request GET \
     --url 'https://api-sandbox.judopay.com/transactions?pageSize=10&offset=integer&sort=
     --header 'accept: application/json' \
     --header 'api-version: 6.23'
```

24.5.3. /transactions/{transactionType}

GET https://api-sandbox.judopay.com/transactions/{transactionType} Try it

Sandbox environment

Return a list of transactions of the given transactionType associated with the given query. Please note: The records returned by this endpoint should not be relied upon for real-time processing.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Path Parameters

- **transactionType** `string` *

Header Parameters

- **Api-Version** `string` *

Query Parameters

- **pageSize** `integer` — 10
- **offset** `integer`
- **sort** `string` — time-descending
- **from** `string`
- **to** `string`
- **yourPaymentReference** `string`

- **yourConsumerReference** `string`

Responses

- **200** Successful response
  - `listTransactionsResponse`
    - **resultCount** `number`
    - **pageSize** `integer`
    - **offset** `integer`
    - **results** `transactionReceiptListResponse[]`
    - **sort** `string`
- **401** Unauthorized
  - `errorResponse`
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`
- **403** Forbidden if token used does not have required permission
  - `errorResponse`
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`

```
curl --request GET \
     --url 'https://api-sandbox.judopay.com/transactions/{transactionType}?pageSize=10&o
     --header 'accept: application/json' \
     --header 'api-version: 6.23'
```

24.6. Payment Sessions

24.6.1. Create a generic payment session

POST https://api-sandbox.judopay.com/paymentsession Try it

Sandbox environment

Create a payment session without specifying a payment operation. webPaymentOperation will be 0.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

> Api-Version `string` *

Body Parameters

- **body** `paymentSessionRequestForPreAuths` * — All Of: paymentSession...
  - **judoId** `string` *
  - **yourConsumerReference** `string` *
  - **yourPaymentReference** `string` *
  - **yourPaymentMetaData** `object` *
  - **currency** `string` *
  - **amount** `number<float>`
  - **cardAddress** `cardAddressWebPaymentRequest` — All Of: cardAddressReq...
  - **expiryDate** `string<date-time>` — mm/dd/yyyy, --:-- -
  - **isPayByLink** `boolean`
  - **isJudoAccept** `boolean`
  - **successUrl** `string`
  - **cancelUrl** `string`
  - **emailAddress** `string`
  - **mobileNumber** `string`
  - **phoneCountryCode** `string` — 44
  - **threeDSecure** `object`
  - **hideBillingInfo** `boolean`
  - **hideReviewInfo** `boolean`
  - **disableNetworkTokenisation** `boolean`
  - **primaryAccountDetails** `object`

- **allowIncrement** `boolean`

Responses

- **200** Successful response
  - `paymentSessionResponse`
    - **expiryDate** `string<date-time>`
    - **reference** `string`
- **400** Bad request (Api-Version header missing, or request body attributes invalid)
- **401** Unauthorized
  - `errorResponse`
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`
- **403** Forbidden if token used does not have required permission
  - `errorResponse`
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`

```
curl --request POST \
     --url https://api-sandbox.judopay.com/paymentsession \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "judoId": "100100100",
     "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
     "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
     "yourPaymentMetaData": "{\"internalLocationRef\":\"Example\",\"internalId\":99}",
     "currency": "GBP",
     "cardAddress": {
     "address1": "CardHolder House",
     "address2": "1 CardHolder Street",
     "town": "CardHolder Town",
     "postCode": "AB1 2CD",
     "state": "FL",
     "countryCode": "826"
     },
     "phoneCountryCode": "44",
     "threeDSecure": {
     "challengeRequestIndicator": "noPreference"
     },
     "hideBillingInfo": true,
     "hideReviewInfo": true,
     "primaryAccountDetails": {
     "name": "Doe",
     "accountNumber": "12345678",
     "dateOfBirth": "1980-01-31",
     "postCode": "AB1 2CD"
     }
     }'
```

```
// Successful response
{
  "expiryDate": "2026-02-05T16:28:32.8596+00:00",
  "reference": "5QcAAAQAAAAPAAAACAAAABtGgvhBrF9BHTN7nqn1e0J4hVVmi-y27dGPjWBMtls3Gj_XDg"
}
```

24.6.2. Cancel an open payment session

PUT https://api-sandbox.judopay.com/paymentsession/{reference}/canc Try it

Sandbox environment

Update the status of an Open payment session to Cancelled, preventing it from being used in the future.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Path Parameters

reference string *

Header Parameters

Api-Version string *

Body Parameters

body object *

Responses

200 Successful response

paymentSessionCancelResponse

reference string

status string

400 Bad request (Api-Version header missing, or payment session is not in Open status)

errorResponse

- requestId string
- message string
- code integer
- category integer

401 Unauthorized

errorResponse

- requestId string
- message string
- code integer
- category integer

403 Forbidden if token used does not have required permission

errorResponse

- requestId string
- message string
- code integer
- category integer

404 Payment session reference not found

errorResponse

- requestId string
- message string
- code integer
- category integer

```
curl --request PUT \
     --url https://api-sandbox.judopay.com/paymentsession/{reference}/cancel \
     --header 'accept: application/json' \
     --header 'api-version: 6.23'
```

```
// Successful response
{
  "reference": "5QcAAAQAAAAPAAAACAAAABtGgvhBrF9BHTN7nqn1e0J4hVVmi-y27dGPjWBMtls3Gj_XDg",
  "status": "Cancelled"
}
```

24.6.3. Retrieve details of a completed payment session

GET https://api-sandbox.judopay.com/transactions/{receiptId}/webpaym Try it

Sandbox environment

Retrieve details of a completed payment session matching the given receiptId.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Path Parameters

> **receiptId** `string` *

Header Parameters

> **Api-Version** `string` *

Responses

> ● **200** Successful response

paymentSessionHistoricResponse

- amount number<float>
- cardAddress object

All Of
cardAddressRes...

- clientIpAddress string
- clientUserAgent string
- companyName string
- currency string
- expiryDate string<date-time>
- judoId string
- paymentCancelUrl string
- paymentSuccessUrl string
- reference string
- allowedCardTyes integer[]
- response object
- status string<Open | Success | Expired …
- transactionType string<Payment | PreAuth…
- yourConsumerReference string
- yourPaymentMetaData object
- yourPaymentReference string
- receipt transactionReceiptHistoricRespo…
- portalUserRecId integer
- webPaymentOperation integer

- **isPayByLink** `boolean`
- **isJudoAccept** `boolean`
- **isThreeDSecureTwo** `boolean`
- **mobileNumber** `string`
- **phoneCountryCode** `string`
- **emailAddress** `string`
- **noOfAuthAttempts** `integer`
- **shortReference** `string`

- **400** Bad request (Api-Version header missing)
- **401** Unauthorized

`errorResponse`

- **requestId** `string`
- **message** `string`
- **code** `integer`
- **category** `integer`

- **403** Forbidden if token used does not have required permission

`errorResponse`

- **requestId** `string`
- **message** `string`
- **code** `integer`
- **category** `integer`

- **404** ReceiptId not found, or not associated with a payment session, or doesn't match authentication details.

```
curl --request GET \
     --url https://api-sandbox.judopay.com/transactions/{receiptId}/webpayment \
     --header 'accept: application/json' \
     --header 'api-version: 6.23'
```

```
// Successful response
{
  "amount": 10.99,
  "cardAddress": {},
  "clientIpAddress": "1.2.3.4",
  "clientUserAgent": "Mozilla/5.0",
  "companyName": "Test Merchant",
  "currency": "GBP",
  "expiryDate": "2026-02-05T16:28:32.8596+00:00",
  "judoId": "100100100",
  "paymentCancelUrl": "https://my.web.site/cancel",
  "paymentSuccessUrl": "https://my.web.site/success",
  "reference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7JXb2xeQRqyYw",
  "allowedCardTyes": [
    0
  ],
  "response": {
    "postUrl": "https://pay-sandbox.judopay.com/v2",
    "reference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7JXb2xeQRqyYw"
  },
  "status": "Success",
  "transactionType": "PreAuth",
  "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
  "yourPaymentMetaData": {},
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "receipt": {
    "receiptId": "1001131610340495360",
    "originalReceiptId": "1001124307998347264",
    "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
    "type": "PreAuth",
    "createdAt": "2025-02-05T16:28:32.8596+00:00",
    "result": "Success",
    "message": "AuthCode: 123456",
    "judoId": 100100100,
    "merchantName": "Test Merchant",
    "appearsOnStatementAs": "APL*/TestMerchant       ",
    "originalAmount": "10.99",
    "amountCollected": "1.99",
    "netAmount": "1.00",
    "amount": "1.00",
    "currency": "GBP",
    "recurringPaymentType": "MIT",
    "acquirerTransactionId": "33666277256892153705",
```

```
"externalBankResponseCode": "A",
"authCode": "123456",
"postCodeCheckResult": "Passed",
"walletType": 1,
"acquirer": "MyBank",
"webPaymentReference": "5QcAAAMAAAASAAAADAAAAHaKm0p0p-Ew6VrBhPxjxvMLinJOMZEJWG_K7kJ5
"noOfAuthAttempts": 1,
"paymentNetworkTransactionId": "123456789012345",
"allowIncrement": true,
"isIncrementalAuth": true,
"disableNetworkTokenisation": true,
"cardDetails": {
  "cardLastfour": "1234",
  "endDate": "1225",
  "cardToken": "Ck3AeNlBfjvs9d61MNZiG0gtCvijqvKr",
  "cardType": 2,
  "startDate": "0121",
  "cardScheme": "Mastercard",
  "cardFunding": "Credit",
```

24.6.4. Return a list of payment sessions

GET https://api-sandbox.judopay.com/webpayments Try it

Sandbox environment

Return a list of payment sessions matching the given query.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

- Api-Version `string` *

Query Parameters

- pageSize `integer` — 10
- offset `integer`
- sort `string` — time-descending

Responses

- 200 Successful response
  - `listPaymentSessionsResponse`
    - resultCount `number`
    - pageSize `integer`
    - offset `integer`
    - results `paymentSessionHistoricListRespon…`

- **400** Bad request (Api-Version header missing)
- **401** Unauthorized
  - errorResponse
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`
- **403** Forbidden if token used does not have required permission
  - errorResponse
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`

```
curl --request GET \
     --url 'https://api-sandbox.judopay.com/webpayments?pageSize=10&offset=integer&sort=
     --header 'accept: application/json' \
     --header 'api-version: 6.23'
```

```
// Successful response
{
  "resultCount": 135,
  "pageSize": 10,
  "offset": 0,
  "results": [
    {
      "amount": 10.99,
      "cardAddress": {},
      "clientIpAddress": "1.2.3.4",
      "clientUserAgent": "Mozilla/5.0",
      "companyName": "Test Merchant",
      "currency": "GBP",
      "expiryDate": "2026-02-05T16:28:32.8596+00:00",
      "judoId": "100100100",
      "paymentCancelUrl": "https://my.web.site/cancel",
      "paymentSuccessUrl": "https://my.web.site/success",
      "reference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7JXb2xeQRqy
      "allowedCardTyes": [
        0
      ],
      "status": "Open",
      "transactionType": "PreAuth",
      "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
      "yourPaymentMetaData": {},
      "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
      "webPaymentOperation": 0,
      "isPayByLink": true,
      "isJudoAccept": false,
      "tradingName": "Test Merchant"
    }
  ]
}
```

24.6.5. Retrieve the details of a single payment session

GET https://api-sandbox.judopay.com/webpayments/{reference} Try it

Sandbox environment

Return the details of a single payment session with the given reference.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Path Parameters

> reference `string` *

Header Parameters

> Api-Version `string` *

Responses

200 Successful response

paymentSessionHistoricResponse

- amount number<float>
- cardAddress object

All Of
cardAddressRes...

- clientIpAddress string
- clientUserAgent string
- companyName string
- currency string
- expiryDate string<date-time>
- judoId string
- paymentCancelUrl string
- paymentSuccessUrl string
- reference string
- allowedCardTyes integer[]
- response object
- status string<Open | Success | Expired …
- transactionType string<Payment | PreAuth…
- yourConsumerReference string
- yourPaymentMetaData object
- yourPaymentReference string
- receipt transactionReceiptHistoricRespo…
- portalUserRecId integer
- webPaymentOperation integer

- **isPayByLink** `boolean`
- **isJudoAccept** `boolean`
- **isThreeDSecureTwo** `boolean`
- **mobileNumber** `string`
- **phoneCountryCode** `string`
- **emailAddress** `string`
- **noOfAuthAttempts** `integer`
- **shortReference** `string`

● **400** Bad request (Api-Version header missing)

● **401** Unauthorized

`errorResponse`

- **requestId** `string`
- **message** `string`
- **code** `integer`
- **category** `integer`

● **403** Forbidden if token used does not have required permission

`errorResponse`

- **requestId** `string`
- **message** `string`
- **code** `integer`
- **category** `integer`

● **404** Reference not found, or doesn't match authentication details.

```
curl --request GET \
     --url https://api-sandbox.judopay.com/webpayments/{reference} \
     --header 'accept: application/json' \
     --header 'api-version: 6.23'
```

```
// Successful response
{
  "amount": 10.99,
  "cardAddress": {},
  "clientIpAddress": "1.2.3.4",
  "clientUserAgent": "Mozilla/5.0",
  "companyName": "Test Merchant",
  "currency": "GBP",
  "expiryDate": "2026-02-05T16:28:32.8596+00:00",
  "judoId": "100100100",
  "paymentCancelUrl": "https://my.web.site/cancel",
  "paymentSuccessUrl": "https://my.web.site/success",
  "reference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7JXb2xeQRqyYw",
  "allowedCardTyes": [
    0
  ],
  "response": {
    "postUrl": "https://pay-sandbox.judopay.com/v2",
    "reference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7JXb2xeQRqyYw"
  },
  "status": "Success",
  "transactionType": "PreAuth",
  "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
  "yourPaymentMetaData": {},
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "receipt": {
    "receiptId": "1001131610340495360",
    "originalReceiptId": "1001124307998347264",
    "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
    "type": "PreAuth",
    "createdAt": "2025-02-05T16:28:32.8596+00:00",
    "result": "Success",
    "message": "AuthCode: 123456",
    "judoId": 100100100,
    "merchantName": "Test Merchant",
    "appearsOnStatementAs": "APL*/TestMerchant       ",
    "originalAmount": "10.99",
    "amountCollected": "1.99",
    "netAmount": "1.00",
    "amount": "1.00",
    "currency": "GBP",
    "recurringPaymentType": "MIT",
    "acquirerTransactionId": "33666277256892153705",
```

```
"externalBankResponseCode": "A",
"authCode": "123456",
"postCodeCheckResult": "Passed",
"walletType": 1,
"acquirer": "MyBank",
"webPaymentReference": "5QcAAAMAAAASAAAADAAAAHaKm0p0p-Ew6VrBhPxjxvMLinJOMZEJWG_K7kJ5
"noOfAuthAttempts": 1,
"paymentNetworkTransactionId": "123456789012345",
"allowIncrement": true,
"isIncrementalAuth": true,
"disableNetworkTokenisation": true,
"cardDetails": {
  "cardLastfour": "1234",
  "endDate": "1225",
  "cardToken": "Ck3AeNlBfjvs9d61MNZiG0gtCvijqvKr",
  "cardType": 2,
  "startDate": "0121",
  "cardScheme": "Mastercard",
  "cardFunding": "Credit",
```

24.6.6. Retrieve the details of a single payment session for a payment

GET https://api-sandbox.judopay.com/webpayments/payments/{referenc Try it

Sandbox environment

Return the details of a single payment session associated with a payment with the given reference.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Path Parameters

reference `string` *

Header Parameters

Api-Version `string` *

Responses

200 Successful response

paymentSessionHistoricResponse

- amount number<float>
- cardAddress object — All Of cardAddressRes...
- clientIpAddress string
- clientUserAgent string
- companyName string
- currency string
- expiryDate string<date-time>
- judoId string
- paymentCancelUrl string
- paymentSuccessUrl string
- reference string
- allowedCardTyes integer[]
- response object
- status string<Open | Success | Expired …
- transactionType string<Payment | PreAuth…
- yourConsumerReference string
- yourPaymentMetaData object
- yourPaymentReference string
- receipt transactionReceiptHistoricRespo…
- portalUserRecId integer
- webPaymentOperation integer

- **isPayByLink** `boolean`
- **isJudoAccept** `boolean`
- **isThreeDSecureTwo** `boolean`
- **mobileNumber** `string`
- **phoneCountryCode** `string`
- **emailAddress** `string`
- **noOfAuthAttempts** `integer`
- **shortReference** `string`

- **400** Bad request (Api-Version header missing)
- **401** Unauthorized

`errorResponse`

- **requestId** `string`
- **message** `string`
- **code** `integer`
- **category** `integer`

- **403** Forbidden if token used does not have required permission

`errorResponse`

- **requestId** `string`
- **message** `string`
- **code** `integer`
- **category** `integer`

- **404** Reference not found, or doesn't match authentication details, or not for a payment

```
curl --request GET \
     --url https://api-sandbox.judopay.com/webpayments/payments/{reference} \
     --header 'accept: application/json' \
     --header 'api-version: 6.23'
```

```
// Successful response
{
  "amount": 10.99,
  "cardAddress": {},
  "clientIpAddress": "1.2.3.4",
  "clientUserAgent": "Mozilla/5.0",
  "companyName": "Test Merchant",
  "currency": "GBP",
  "expiryDate": "2026-02-05T16:28:32.8596+00:00",
  "judoId": "100100100",
  "paymentCancelUrl": "https://my.web.site/cancel",
  "paymentSuccessUrl": "https://my.web.site/success",
  "reference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7JXb2xeQRqyYw",
  "allowedCardTyes": [
    0
  ],
  "response": {
    "postUrl": "https://pay-sandbox.judopay.com/v2",
    "reference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7JXb2xeQRqyYw"
  },
  "status": "Success",
  "transactionType": "PreAuth",
  "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
  "yourPaymentMetaData": {},
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "receipt": {
    "receiptId": "1001131610340495360",
    "originalReceiptId": "1001124307998347264",
    "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
    "type": "PreAuth",
    "createdAt": "2025-02-05T16:28:32.8596+00:00",
    "result": "Success",
    "message": "AuthCode: 123456",
    "judoId": 100100100,
    "merchantName": "Test Merchant",
    "appearsOnStatementAs": "APL*/TestMerchant       ",
    "originalAmount": "10.99",
    "amountCollected": "1.99",
    "netAmount": "1.00",
    "amount": "1.00",
    "currency": "GBP",
    "recurringPaymentType": "MIT",
    "acquirerTransactionId": "33666277256892153705",
```

```
"externalBankResponseCode": "A",
"authCode": "123456",
"postCodeCheckResult": "Passed",
"walletType": 1,
"acquirer": "MyBank",
"webPaymentReference": "5QcAAAMAAAASAAAADAAAAHaKm0p0p-Ew6VrBhPxjxvMLinJOMZEJWG_K7kJ5
"noOfAuthAttempts": 1,
"paymentNetworkTransactionId": "123456789012345",
"allowIncrement": true,
"isIncrementalAuth": true,
"disableNetworkTokenisation": true,
"cardDetails": {
  "cardLastfour": "1234",
  "endDate": "1225",
  "cardToken": "Ck3AeNlBfjvs9d61MNZiG0gtCvijqvKr",
  "cardType": 2,
  "startDate": "0121",
  "cardScheme": "Mastercard",
  "cardFunding": "Credit",
```

24.6.7. Retrieve the details of a single payment session for a preauth

GET https://api-sandbox.judopay.com/webpayments/preauths/{reference

Try it

Sandbox environment

Return the details of a single payment session associated with a pre-authorisation with the given reference.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Path Parameters

reference `string` *

Header Parameters

Api-Version `string` *

Responses

200 Successful response

paymentSessionHistoricResponse

- amount number<float>
- cardAddress object — All Of cardAddressRes...
- clientIpAddress string
- clientUserAgent string
- companyName string
- currency string
- expiryDate string<date-time>
- judoId string
- paymentCancelUrl string
- paymentSuccessUrl string
- reference string
- allowedCardTyes integer[]
- response object
- status string<Open | Success | Expired …
- transactionType string<Payment | PreAuth…
- yourConsumerReference string
- yourPaymentMetaData object
- yourPaymentReference string
- receipt transactionReceiptHistoricRespo…
- portalUserRecId integer
- webPaymentOperation integer

- isPayByLink `boolean`
- isJudoAccept `boolean`
- isThreeDSecureTwo `boolean`
- mobileNumber `string`
- phoneCountryCode `string`
- emailAddress `string`
- noOfAuthAttempts `integer`
- shortReference `string`

- 400 Bad request (Api-Version header missing)
- 401 Unauthorized

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

- 403 Forbidden if token used does not have required permission

`errorResponse`

- requestId `string`
- message `string`
- code `integer`
- category `integer`

- 404 Reference not found, or doesn't match authentication details, or not for a preauth

```
curl --request GET \
     --url https://api-sandbox.judopay.com/webpayments/preauths/{reference} \
     --header 'accept: application/json' \
     --header 'api-version: 6.23'
```

```
// Successful response
{
  "amount": 10.99,
  "cardAddress": {},
  "clientIpAddress": "1.2.3.4",
  "clientUserAgent": "Mozilla/5.0",
  "companyName": "Test Merchant",
  "currency": "GBP",
  "expiryDate": "2026-02-05T16:28:32.8596+00:00",
  "judoId": "100100100",
  "paymentCancelUrl": "https://my.web.site/cancel",
  "paymentSuccessUrl": "https://my.web.site/success",
  "reference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7JXb2xeQRqyYw",
  "allowedCardTyes": [
    0
  ],
  "response": {
    "postUrl": "https://pay-sandbox.judopay.com/v2",
    "reference": "5QcAAAIAAAAMAAAADwAAAPwlgcQCLU6sLrNhi8kt1vEoVLwiFJc2EKTqT7JXb2xeQRqyYw"
  },
  "status": "Success",
  "transactionType": "PreAuth",
  "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
  "yourPaymentMetaData": {},
  "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
  "receipt": {
    "receiptId": "1001131610340495360",
    "originalReceiptId": "1001124307998347264",
    "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
    "type": "PreAuth",
    "createdAt": "2025-02-05T16:28:32.8596+00:00",
    "result": "Success",
    "message": "AuthCode: 123456",
    "judoId": 100100100,
    "merchantName": "Test Merchant",
    "appearsOnStatementAs": "APL*/TestMerchant       ",
    "originalAmount": "10.99",
    "amountCollected": "1.99",
    "netAmount": "1.00",
    "amount": "1.00",
    "currency": "GBP",
    "recurringPaymentType": "MIT",
    "acquirerTransactionId": "33666277256892153705",
```

```
"externalBankResponseCode": "A",
"authCode": "123456",
"postCodeCheckResult": "Passed",
"walletType": 1,
"acquirer": "MyBank",
"webPaymentReference": "5QcAAAMAAAASAAAADAAAAHaKm0p0p-Ew6VrBhPxjxvMLinJOMZEJWG_K7kJ5
"noOfAuthAttempts": 1,
"paymentNetworkTransactionId": "123456789012345",
"allowIncrement": true,
"isIncrementalAuth": true,
"disableNetworkTokenisation": true,
"cardDetails": {
  "cardLastfour": "1234",
  "endDate": "1225",
  "cardToken": "Ck3AeNlBfjvs9d61MNZiG0gtCvijqvKr",
  "cardType": 2,
  "startDate": "0121",
  "cardScheme": "Mastercard",
  "cardFunding": "Credit",
```

24.6.8. Create a session for check card

POST https://api-sandbox.judopay.com/webpayments/checkcard Try it

Sandbox environment

Create a session for a check card operation (zero amount pre-authorisation). transactionType will be CHECKCARD, webPaymentOperation will be 1.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

> **Api-Version** `string` *

Body Parameters

- **body** `paymentSessionRequest` * — All Of: paymentSession...
  - **judoId** `string` *
  - **yourConsumerReference** `string` *
  - **yourPaymentReference** `string` *
  - **yourPaymentMetaData** `object`
  - **currency** `string`
  - **amount** `number<float>`
  - **cardAddress** `cardAddressWebPaymentRequest` — All Of: cardAddressReq...
  - **expiryDate** `string<date-time>` — mm/dd/yyyy, --:-- -
  - **isPayByLink** `boolean`
  - **isJudoAccept** `boolean`
  - **successUrl** `string`
  - **cancelUrl** `string`
  - **emailAddress** `string`
  - **mobileNumber** `string`
  - **phoneCountryCode** `string` — 44
  - **threeDSecure** `object`
  - **hideBillingInfo** `boolean`
  - **hideReviewInfo** `boolean`
  - **disableNetworkTokenisation** `boolean`
  - **primaryAccountDetails** `object`

Responses

- 200 Successful response
  - createPaymentSessionResponse
    - **payByLinkUrl** `string`
    - **postUrl** `string`
    - **reference** `string`
- 400 Bad request (Api-Version header missing, or request body attributes invalid)
- 401 Unauthorized
  - errorResponse
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`
- 403 Forbidden if token used does not have required permission
  - errorResponse
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`

```
curl --request POST \
     --url https://api-sandbox.judopay.com/webpayments/checkcard \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "judoId": "100100100",
     "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
     "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
     "cardAddress": {
     "postCode": "AB1 2CD"
     },
     "phoneCountryCode": "44",
     "threeDSecure": {
     "challengeRequestIndicator": "noPreference"
     },
     "hideBillingInfo": true,
     "hideReviewInfo": true,
     "primaryAccountDetails": {
     "name": "Doe",
     "accountNumber": "12345678",
     "dateOfBirth": "1980-01-31",
     "postCode": "AB1 2CD"
     }
     }'
```

```
// Successful response
{
  "payByLinkUrl": "https://pay-sandbox.judopay.com/ABC123",
  "postUrl": "https://pay-sandbox.judopay.com/v2",
  "reference": "5QcAAAQAAAAPAAAACAAAABtGgvhBrF9BHTN7nqn1e0J4hVVmi-y27dGPjWBMtls3Gj_XDg"
}
```

24.6.9. Create a session for a payment

POST https://api-sandbox.judopay.com/webpayments/payments Try it

Sandbox environment

Create a session for a payment operation. transactionType will be Payment, webPaymentOperation will be 0.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

> **Api-Version** `string` *

Body Parameters

- **body** `paymentSessionRequest` * — All Of: paymentSession...
  - **judoId** `string` *
  - **yourConsumerReference** `string` *
  - **yourPaymentReference** `string` *
  - **yourPaymentMetaData** `object`
  - **currency** `string`
  - **amount** `number<float>`
  - **cardAddress** `cardAddressWebPaymentRequest` — All Of: cardAddressReq...
  - **expiryDate** `string<date-time>` — mm/dd/yyyy, --:-- -
  - **isPayByLink** `boolean`
  - **isJudoAccept** `boolean`
  - **successUrl** `string`
  - **cancelUrl** `string`
  - **emailAddress** `string`
  - **mobileNumber** `string`
  - **phoneCountryCode** `string` — 44
  - **threeDSecure** `object`
  - **hideBillingInfo** `boolean`
  - **hideReviewInfo** `boolean`
  - **disableNetworkTokenisation** `boolean`
  - **primaryAccountDetails** `object`

Responses

- 200 Successful response
  - createPaymentSessionResponse
    - payByLinkUrl string
    - postUrl string
    - reference string
- 400 Bad request (Api-Version header missing, or request body attributes invalid)
- 401 Unauthorized
  - errorResponse
    - requestId string
    - message string
    - code integer
    - category integer
- 403 Forbidden if token used does not have required permission
  - errorResponse
    - requestId string
    - message string
    - code integer
    - category integer

```
curl --request POST \
     --url https://api-sandbox.judopay.com/webpayments/payments \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "judoId": "100100100",
     "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
     "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
     "cardAddress": {
     "postCode": "AB1 2CD"
     },
     "phoneCountryCode": "44",
     "threeDSecure": {
     "challengeRequestIndicator": "noPreference"
     },
     "hideBillingInfo": true,
     "hideReviewInfo": true,
     "primaryAccountDetails": {
     "name": "Doe",
     "accountNumber": "12345678",
     "dateOfBirth": "1980-01-31",
     "postCode": "AB1 2CD"
     }
     }'
```

```
// Successful response
{
  "payByLinkUrl": "https://pay-sandbox.judopay.com/ABC123",
  "postUrl": "https://pay-sandbox.judopay.com/v2",
  "reference": "5QcAAAQAAAAPAAAACAAAABtGgvhBrF9BHTN7nqn1e0J4hVVmi-y27dGPjWBMtls3Gj_XDg"
}
```

24.6.10. Create a session for a preauth

POST https://api-sandbox.judopay.com/webpayments/preauths Try it

Sandbox environment

Create a session for a pre-authorisation. transactionType will be PREAUTH, webPaymentOperation will be 0.

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

Api-Version `string` *

Body Parameters

- **body** `paymentSessionRequestForPreAuths` * — All Of: paymentSession...
  - **judoId** `string` *
  - **yourConsumerReference** `string` *
  - **yourPaymentReference** `string` *
  - **yourPaymentMetaData** `object` *
  - **currency** `string` *
  - **amount** `number<float>`
  - **cardAddress** `cardAddressWebPaymentRequest` — All Of: cardAddressReq...
  - **expiryDate** `string<date-time>` — mm/dd/yyyy, --:-- -
  - **isPayByLink** `boolean`
  - **isJudoAccept** `boolean`
  - **successUrl** `string`
  - **cancelUrl** `string`
  - **emailAddress** `string`
  - **mobileNumber** `string`
  - **phoneCountryCode** `string` — 44
  - **threeDSecure** `object`
  - **hideBillingInfo** `boolean`
  - **hideReviewInfo** `boolean`
  - **disableNetworkTokenisation** `boolean`
  - **primaryAccountDetails** `object`

- **allowIncrement** `boolean`

Responses

- **200** Successful response
  - `createPaymentSessionResponse`
    - **payByLinkUrl** `string`
    - **postUrl** `string`
    - **reference** `string`
- **400** Bad request (Api-Version header missing, or request body attributes invalid)
- **401** Unauthorized
  - `errorResponse`
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`
- **403** Forbidden if token used does not have required permission
  - `errorResponse`
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`

```
curl --request POST \
     --url https://api-sandbox.judopay.com/webpayments/preauths \
     --header 'accept: application/json' \
     --header 'content-type: application/json' \
     --header 'api-version: 6.23' \
     --data '{
     "judoId": "100100100",
     "yourConsumerReference": "2b45fd3f-cee5-4e7e-874f-28051db65408",
     "yourPaymentReference": "6482c678-cad3-4efd-b081-aeae7a89a134",
     "yourPaymentMetaData": "{\"internalLocationRef\":\"Example\",\"internalId\":99}",
     "currency": "GBP",
     "cardAddress": {
     "address1": "CardHolder House",
     "address2": "1 CardHolder Street",
     "town": "CardHolder Town",
     "postCode": "AB1 2CD",
     "state": "FL",
     "countryCode": "826"
     },
     "phoneCountryCode": "44",
     "threeDSecure": {
     "challengeRequestIndicator": "noPreference"
     },
     "hideBillingInfo": true,
     "hideReviewInfo": true,
     "primaryAccountDetails": {
     "name": "Doe",
     "accountNumber": "12345678",
     "dateOfBirth": "1980-01-31",
     "postCode": "AB1 2CD"
     }
     }'
```

```
// Successful response
{
  "payByLinkUrl": "https://pay-sandbox.judopay.com/ABC123",
  "postUrl": "https://pay-sandbox.judopay.com/v2",
  "reference": "5QcAAAQAAAAPAAAACAAAABtGgvhBrF9BHTN7nqn1e0J4hVVmi-y27dGPjWBMtls3Gj_XDg"
}
```

24.7. Other

24.7.1. /info

GET https://api-sandbox.judopay.com/info Try it

Sandbox environment

Obtain information about service, including latest supported Api-Version

Credentials

HTTP Basic TokenSecretAuth

This is the typical authorisation scenario to use, specify your token in Username and secret in Password.

Sent as Base64 encoded string representing token

in Authorization header.

username : password

Header Parameters

Api-Version `string` *

Responses

200 Successful response

`infoResponse`

version `string`

400 Bad request (Api-Version header missing)

- **401** Unauthorized
  - errorResponse
    - **requestId** `string`
    - **message** `string`
    - **code** `integer`
    - **category** `integer`

```curl
curl --request GET \
     --url https://api-sandbox.judopay.com/info \
     --header 'accept: application/json' \
     --header 'api-version: 6.23'
```

```
// Successful response
{
  "version": "6.23.0.0"
}
```

24.8. Models

24.8.1. Model: allowIncrementRequest

Model Parameters

- `allowIncrementRequest`
  - **allowIncrement** `boolean`


```
{
  "allowIncrement": false
}
```

24.8.2. Model: networkTokenRequest

Model Parameters

- networkTokenRequest
  - disableNetworkTokenisation boolean


```
{
  "disableNetworkTokenisation": false
}
```

24.8.3. Model: applePayRequest

Model Parameters

- applePayRequest
  - pkPayment object *
  - cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...


```
{
  "pkPayment": {
    "token": {
      "paymentMethod": {
        "displayName": "",
        "network": "",
        "type": ""
      },
      "paymentData": {
        "data": "",
        "signature": "",
        "header": {
          "publicKeyHash": "",
          "ephemeralPublicKey": "",
          "transactionId": ""
        },
        "version": ""
      }
    }
  },
  "cardAddress": {}
}
```

24.8.4. Model: cardAddressCommonAttributes

Model Parameters

- cardAddressCommonAttributes
  - address1 string
  - address2 string
  - town string
  - postCode string *
  - state string


```
{
  "address1": "",
  "address2": "",
  "town": "",
  "postCode": "",
  "state": ""
}
```


Card holder address.

24.8.5. Model: cardAddressRequestAttributes

Model Parameters

cardAddressRequestAttributes

All Of
cardAddressCom...

- address1 string
- address2 string
- town string
- postCode string *
- state string
- countryCode string


```
{
  "address1": "",
  "address2": "",
  "town": "",
  "postCode": "",
  "state": "",
  "countryCode": ""
}
```

24.8.6. Model: cardAddressRequest

Model Parameters

cardAddressRequest

cardAddress cardAddressRequestAttributes

All Of
cardAddressCom...


```
{
  "cardAddress": {}
}
```

24.8.7. Model: cardAddressResponseAttributes

Model Parameters

cardAddressResponseAttributes

All Of
cardAddressCom...

- address1 string
- address2 string
- town string
- postCode string *
- state string
- countryCode integer


```
{
  "address1": "",
  "address2": "",
  "town": "",
  "postCode": "",
  "state": "",
  "countryCode": 0
}
```

24.8.8. Model: cardAddressWebPaymentRequest

Model Parameters

cardAddressWebPaymentRequest

All Of
cardAddressReq...

- address1 string
- address2 string
- town string
- postCode string *
- state string
- countryCode string
- cardHolderName string


```
{
  "address1": "",
  "address2": "",
  "town": "",
  "postCode": "",
  "state": "",
  "countryCode": "",
  "cardHolderName": ""
}
```

24.8.9. Model: cardDetailsCv2NotRequiredRequest

Model Parameters

- `cardDetailsCv2NotRequiredRequest`
  - **cardNumber** `string` *
  - **cv2** `string`
  - **expiryDate** `string` *


```
{
  "cardNumber": "",
  "cv2": "",
  "expiryDate": ""
}
```

24.8.10. Model: cardDetailsCv2RequiredRequest

Model Parameters

- `cardDetailsCv2RequiredRequest`
  - **cardNumber** `string` *
  - **cv2** `string` *
  - **expiryDate** `string` *


```
{
  "cardNumber": "",
  "cv2": "",
  "expiryDate": ""
}
```

24.8.11. Model: cardDetailsStartDateRequest

Model Parameters

- cardDetailsStartDateRequest
  - startDate string

```
{
  "startDate": ""
}
```

24.8.12. Model: cardDetailsResponse

Model Parameters

- `cardDetailsResponse`
  - **cardLastfour** `string`
  - **endDate** `string`
  - **cardToken** `string`
  - **cardType** `integer`
  - **startDate** `string`
  - **cardScheme** `string`
  - **cardFunding** `string`
  - **cardCategory** `string`
  - **cardCountry** `string`
  - **bank** `string`
  - **cardHolderName** `string`
  - **ownerType** `string<Personal | Commercial |…`

```
{
  "cardLastfour": "",
  "endDate": "",
  "cardToken": "",
  "cardType": 0,
  "startDate": "",
  "cardScheme": "",
  "cardFunding": "",
  "cardCategory": "",
  "cardCountry": "",
  "bank": "",
  "cardHolderName": "",
  "ownerType": ""
}
```

24.8.13. Model: cardDetailsListResponse

Model Parameters

- `cardDetailsListResponse`
  - **cardLastfour** `string`
  - **endDate** `string`
  - **cardType** `integer`


```
{
  "cardLastfour": "",
  "endDate": "",
  "cardType": 0
}
```

24.8.14. Model: cardTokenPreAuthPaymentRequest

Model Parameters

cardTokenPreAuthPaymentRequest — All Of: object +

- cardToken string *
- cv2 string
- cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...

```
{
  "cardToken": "",
  "cv2": "",
  "cardAddress": {}
}
```

24.8.15. Model: challengeRequestIndicatorRequest

Model Parameters

`challengeRequestIndicatorRequest`

Example

challengePreferred

```
{}
```

Indicates whether a challenge is requested for this transaction (this may be over-ruled by the issuer). Should not be specified in the same request as scaExemption

24.8.16. Model: checkCardRequestPan

Model Parameters

checkCardRequestPan — All Of: cardDetailsCv2N...

- cardNumber string *
- cv2 string
- expiryDate string *
- startDate string
- yourConsumerReference string *
- yourPaymentReference string
- clientDetails object
- judoId string
- currency string
- initialRecurringPayment boolean
- disableNetworkTokenisation boolean
- cardHolderName string
- emailAddress string
- shippingAddress object
- mobileNumber string
- phoneCountryCode string
- threeDSecure object
- threeDSecureMpi object
- cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...
- primaryAccountDetails object

24.8.17. Model: collectionRequest

Model Parameters

- collectionRequest
  - receiptId string *
  - amount number<float>
  - yourPaymentMetaData object
  - yourPaymentReference string *

```
{
  "receiptId": "",
  "amount": "",
  "yourPaymentMetaData": {},
  "yourPaymentReference": ""
}
```

24.8.18. Model: complete3ds2Request

Model Parameters

complete3ds2Request

All Of
object +

- cv2 string
- version string *
- primaryAccountDetails object


```
{
  "cv2": "",
  "version": "",
  "primaryAccountDetails": {
    "name": "",
    "accountNumber": "",
    "dateOfBirth": "",
    "postCode": ""
  }
}
```

24.8.19. Model: consumerResponse

Model Parameters

- `consumerResponse`
  - **yourConsumerReference** `string`

```
{
  "yourConsumerReference": ""
}
```

24.8.20. Model: createPaymentSessionResponse

Model Parameters

- `createPaymentSessionResponse`
  - **payByLinkUrl** `string`
  - **postUrl** `string`
  - **reference** `string`


```
{
  "payByLinkUrl": "",
  "postUrl": "",
  "reference": ""
}
```

24.8.21. Model: deviceResponse

Model Parameters

- `deviceResponse`
  - **identifier** `string`

```
{
  "identifier": ""
}
```

[Conditional] If clientDetails was provided containing a kdeviceid

24.8.22. Model: errorResponse

Model Parameters

- errorResponse
  - requestId string
  - message string
  - code integer
  - category integer


```
{
  "requestId": "",
  "message": "",
  "code": 0,
  "category": 0
}
```

24.8.23. Model: googlePayRequest

Model Parameters

googlePayRequest

- googlePayWallet object *
- cardAddress cardAddressRequestAttributes — All Of cardAddressCom...


```
{
  "googlePayWallet": {
    "token": "",
    "cardNetwork": "",
    "cardDetails": ""
  },
  "cardAddress": {}
}
```

24.8.24. Model: incrementAuthRequest

Model Parameters

- incrementAuthRequest
  - receiptId string *
  - amount number<float> *
  - yourPaymentMetaData object
  - yourPaymentReference string *


```
{
  "receiptId": "",
  "amount": "",
  "yourPaymentMetaData": {},
  "yourPaymentReference": ""
}
```

24.8.25. Model: infoResponse

Model Parameters

- infoResponse
  - version string


```
{
  "version": ""
}
```

24.8.26. Model: listPaymentSessionsResponse

Model Parameters

- `listPaymentSessionsResponse`
  - **resultCount** `number`
  - **pageSize** `integer`
  - **offset** `integer`
  - **results** `paymentSessionHistoricListResponse…`

```
{
  "resultCount": 0,
  "pageSize": 0,
  "offset": 0,
  "results": [
    {
      "amount": "",
      "cardAddress": {},
      "clientIpAddress": "",
      "clientUserAgent": "",
      "companyName": "",
      "currency": "",
      "expiryDate": "",
      "judoId": "",
      "paymentCancelUrl": "",
      "paymentSuccessUrl": "",
      "reference": "",
      "allowedCardTyes": [
        0
      ],
      "status": "",
      "transactionType": "",
      "yourConsumerReference": "",
      "yourPaymentMetaData": {},
      "yourPaymentReference": "",
      "webPaymentOperation": 0,
      "isPayByLink": false,
      "isJudoAccept": false,
      "tradingName": ""
    }
  ]
}
```

24.8.27. Model: listTransactionsResponse

Model Parameters

- `listTransactionsResponse`
  - **resultCount** `number`
  - **pageSize** `integer`
  - **offset** `integer`
  - **results** `transactionReceiptListResponse[]`
  - **sort** `string`

24.8.28. Model: paymentRequestApplePay

Model Parameters

- paymentRequestApplePay — All Of: applePayRequest +
  - pkPayment `object` *
  - cardAddress `cardAddressRequestAttributes` — All Of: cardAddressCom...
  - judoId `string` *
  - yourConsumerReference `string` *
  - yourPaymentReference `string` *
  - yourPaymentMetaData `object` *
  - currency `string` *
  - amount `number<float>` *
  - webPaymentReference `string`
  - clientDetails `object`
  - initialRecurringPayment `boolean`
  - primaryAccountDetails `object`

```
{
  "pkPayment": {
    "token": {
      "paymentMethod": {
        "displayName": "",
        "network": "",
        "type": ""
      },
      "paymentData": {
        "data": "",
        "signature": "",
        "header": {
          "publicKeyHash": "",
          "ephemeralPublicKey": "",
          "transactionId": ""
        },
        "version": ""
      }
    }
  },
  "cardAddress": {},
  "judoId": "",
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "yourPaymentMetaData": {},
  "currency": "",
  "amount": "",
  "webPaymentReference": "",
  "clientDetails": {
    "key": "",
    "value": ""
  },
  "initialRecurringPayment": false,
  "primaryAccountDetails": {
    "name": "",
    "accountNumber": "",
    "dateOfBirth": "",
    "postCode": ""
  }
}
```

24.8.29. Model: paymentRequestCardToken

Model Parameters

- paymentRequestCardToken — All Of: cardTokenPreAut...
  - cardToken string *
  - cv2 string
  - cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...
  - judoId string *
  - yourConsumerReference string *
  - yourPaymentReference string *
  - yourPaymentMetaData object *
  - currency string *
  - amount number<float> *
  - webPaymentReference string
  - clientDetails object
  - initialRecurringPayment boolean
  - primaryAccountDetails object
  - acceptHeaders string
  - userAgent string
  - cardHolderName string
  - emailAddress string
  - shippingAddress object
  - mobileNumber string
  - phoneCountryCode string

- **threeDSecure** `object`
- **threeDSecureMpi** `object`
- **recurringPayment** `boolean`
- **recurringPaymentType** `string<RECURRING | MI…`
- **relatedReceiptId** `string`
- **relatedPaymentNetworkTransactionId** `string`
- **disableNetworkTokenisation** `boolean`

```
{
  "cardToken": "",
  "cv2": "",
  "cardAddress": {},
  "judoId": "",
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "yourPaymentMetaData": {},
  "currency": "",
  "amount": ""
```

24.8.30. Model: paymentRequestGooglePay

Model Parameters

- paymentRequestGooglePay — All Of: googlePayReque...
  - googlePayWallet object *
  - cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...
  - judoId string *
  - yourConsumerReference string *
  - yourPaymentReference string *
  - yourPaymentMetaData object *
  - currency string *
  - amount number<float> *
  - webPaymentReference string
  - clientDetails object
  - initialRecurringPayment boolean
  - primaryAccountDetails object
  - acceptHeaders string
  - userAgent string
  - cardHolderName string
  - emailAddress string
  - shippingAddress object
  - mobileNumber string
  - phoneCountryCode string
  - threeDSecure object

> **threeDSecureMpi** `object`

```
{
  "googlePayWallet": {
    "token": "",
    "cardNetwork": "",
    "cardDetails": ""
  },
  "cardAddress": {},
  "judoId": ""
```

24.8.31. Model: paymentRequestPan

Model Parameters

- paymentRequestPan — All Of: cardDetailsCv2N...
  - cardNumber string *
  - cv2 string
  - expiryDate string *
  - startDate string
  - cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...
  - judoId string *
  - yourConsumerReference string *
  - yourPaymentReference string *
  - yourPaymentMetaData object *
  - currency string *
  - amount number<float> *
  - webPaymentReference string
  - clientDetails object
  - initialRecurringPayment boolean
  - primaryAccountDetails object
  - acceptHeaders string
  - userAgent string
  - cardHolderName string
  - emailAddress string
  - shippingAddress object

- **mobileNumber** `string`
- **phoneCountryCode** `string`
- **threeDSecure** `object`
- **threeDSecureMpi** `object`
- **recurringPayment** `boolean`
- **recurringPaymentType** `string<RECURRING | MI…`
- **relatedReceiptId** `string`
- **relatedPaymentNetworkTransactionId** `string`
- **disableNetworkTokenisation** `boolean`

```
{
  "cardNumber": "",
  "cv2": "",
  "expiryDate": "",
  "startDate": "",
  "cardAddress": {},
  "judoId": "",
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "yourPaymentMetaData": {},
  "currency": "",
  "amount": ""
```

24.8.32. Model: paymentSessionHistoricResponse

Model Parameters

paymentSessionHistoricResponse

- amount number<float>
- cardAddress object — All Of cardAddressRes...
- clientIpAddress string
- clientUserAgent string
- companyName string
- currency string
- expiryDate string<date-time>
- judoId string
- paymentCancelUrl string
- paymentSuccessUrl string
- reference string
- allowedCardTyes integer[]
- response object
- status string<Open | Success | Expired | …
- transactionType string<Payment | PreAuth |…
- yourConsumerReference string
- yourPaymentMetaData object
- yourPaymentReference string
- receipt transactionReceiptHistoricResponse
- portalUserRecId integer

- webPaymentOperation `integer`
- isPayByLink `boolean`
- isJudoAccept `boolean`
- isThreeDSecureTwo `boolean`
- mobileNumber `string`
- phoneCountryCode `string`
- emailAddress `string`
- noOfAuthAttempts `integer`
- shortReference `string`

```
{
  "amount": "",
  "cardAddress": {},
  "clientIpAddress": "",
  "clientUserAgent": "",
  "companyName": "",
  "currency": "",
  "expiryDate": "",
  "judoId": "",
  "paymentCancelUrl": "",
  "paymentSuccessUrl": "",
  "reference": "",
  "allowedCardTyes": [
    0
  ],
  "response": {
    "postUrl": "",
    "reference": ""
  },
  "status": "",
  "transactionType": "",
  "yourConsumerReference": "",
  "yourPaymentMetaData": {},
  "yourPaymentReference": "",
  "receipt": {
    "receiptId": "",
    "originalReceiptId": "",
    "yourPaymentReference": "",
    "type": "",
    "createdAt": "",
    "result": "",
    "message": "",
    "judoId": 0,
    "merchantName": "",
    "appearsOnStatementAs": "",
    "originalAmount": "",
    "amountCollected": "",
    "netAmount": "",
    "amount": "",
    "currency": "",
    "recurringPaymentType": "",
    "acquirerTransactionId": "",
    "externalBankResponseCode": "",
```

```
"authCode": "",
"postCodeCheckResult": "",
"walletType": 0,
"acquirer": "",
"webPaymentReference": "",
"noOfAuthAttempts": 0,
"paymentNetworkTransactionId": "",
"allowIncrement": false,
"isIncrementalAuth": false,
"disableNetworkTokenisation": false,
"cardDetails": {
  "cardLastfour": "",
  "endDate": "",
  "cardToken": "",
  "cardType": 0,
  "startDate": "",
  "cardScheme": "",
  "cardFunding": "",
```

24.8.33. Model: paymentSessionHistoricListResponse

Model Parameters

paymentSessionHistoricListResponse

- amount number<float>
- cardAddress object — All Of cardAddressRes...
- clientIpAddress string
- clientUserAgent string
- companyName string
- currency string
- expiryDate string<date-time>
- judoId string
- paymentCancelUrl string
- paymentSuccessUrl string
- reference string
- allowedCardTyes integer[]
- status string<Open | Success | Expired | …
- transactionType string<Payment | PreAuth |…
- yourConsumerReference string
- yourPaymentMetaData object
- yourPaymentReference string
- webPaymentOperation integer
- isPayByLink boolean
- isJudoAccept boolean

> **tradingName** `string`


```
{
  "amount": "",
  "cardAddress": {},
  "clientIpAddress": "",
  "clientUserAgent": "",
  "companyName": "",
  "currency": "",
  "expiryDate": "",
  "judoId": "",
  "paymentCancelUrl": "",
  "paymentSuccessUrl": "",
  "reference": "",
  "allowedCardTyes": [
    0
  ],
  "status": "",
  "transactionType": "",
  "yourConsumerReference": "",
  "yourPaymentMetaData": {},
  "yourPaymentReference": "",
  "webPaymentOperation": 0,
  "isPayByLink": false,
  "isJudoAccept": false,
  "tradingName": ""
}
```

24.8.34. Model: paymentSessionRequest

Model Parameters

- paymentSessionRequest — All Of: paymentSession...
  - judoId `string` *
  - yourConsumerReference `string` *
  - yourPaymentReference `string` *
  - yourPaymentMetaData `object`
  - currency `string`
  - amount `number<float>`
  - cardAddress `cardAddressWebPaymentRequest` — All Of: cardAddressReq...
  - expiryDate `string<date-time>`
  - isPayByLink `boolean`
  - isJudoAccept `boolean`
  - successUrl `string`
  - cancelUrl `string`
  - emailAddress `string`
  - mobileNumber `string`
  - phoneCountryCode `string`
  - threeDSecure `object`
  - hideBillingInfo `boolean`
  - hideReviewInfo `boolean`
  - disableNetworkTokenisation `boolean`
  - primaryAccountDetails `object`

```
{
  "judoId": "",
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "yourPaymentMetaData": {},
  "currency": "",
  "amount": "",
  "cardAddress": {},
  "expiryDate": "",
  "isPayByLink": false,
  "isJudoAccept": false,
  "successUrl": "",
  "cancelUrl": "",
  "emailAddress": "",
  "mobileNumber": "",
  "phoneCountryCode": "44",
  "threeDSecure": {
    "challengeRequestIndicator": "noPreference",
    "scaExemption": ""
  },
  "hideBillingInfo": true,
  "hideReviewInfo": true,
  "disableNetworkTokenisation": false,
  "primaryAccountDetails": {
    "name": "",
    "accountNumber": "",
    "dateOfBirth": "",
    "postCode": ""
  }
}
```

24.8.35. Model: paymentSessionRequestForPreAuths

Model Parameters

- paymentSessionRequestForPreAuths — All Of: paymentSession...
  - judoId `string` *
  - yourConsumerReference `string` *
  - yourPaymentReference `string` *
  - yourPaymentMetaData `object` *
  - currency `string` *
  - amount `number<float>`
  - cardAddress `cardAddressWebPaymentRequest` — All Of: cardAddressReq...
  - expiryDate `string<date-time>`
  - isPayByLink `boolean`
  - isJudoAccept `boolean`
  - successUrl `string`
  - cancelUrl `string`
  - emailAddress `string`
  - mobileNumber `string`
  - phoneCountryCode `string`
  - threeDSecure `object`
  - hideBillingInfo `boolean`
  - hideReviewInfo `boolean`
  - disableNetworkTokenisation `boolean`
  - primaryAccountDetails `object`

› **allowIncrement** `boolean`


```
{
  "judoId": "",
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "yourPaymentMetaData": {},
  "currency": "",
  "amount": "",
  "cardAddress": {},
  "expiryDate": "",
  "isPayByLink": false,
  "isJudoAccept": false,
  "successUrl": "",
  "cancelUrl": "",
  "emailAddress": "",
  "mobileNumber": "",
  "phoneCountryCode": "44",
  "threeDSecure": {
    "challengeRequestIndicator": "noPreference",
    "scaExemption": ""
  },
  "hideBillingInfo": true,
  "hideReviewInfo": true,
  "disableNetworkTokenisation": false,
  "primaryAccountDetails": {
    "name": "",
    "accountNumber": "",
    "dateOfBirth": "",
    "postCode": ""
  },
  "allowIncrement": false
}
```

24.8.36. Model: paymentSessionRequestCommon

Model Parameters

paymentSessionRequestCommon

All Of
object +

- judoId string *
- yourConsumerReference string *
- yourPaymentReference string *
- yourPaymentMetaData object
- currency string


```
{
  "judoId": "",
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "yourPaymentMetaData": {},
  "currency": ""
}
```

24.8.37. Model: paymentSessionResponse

Model Parameters

- paymentSessionResponse
  - expiryDate `string<date-time>`
  - reference `string`


```
{
  "expiryDate": "",
  "reference": ""
}
```

24.8.38. Model: paymentSessionCancelResponse

Model Parameters

- paymentSessionCancelResponse
  - reference string
  - status string


```
{
  "reference": "",
  "status": ""
}
```

24.8.39. Model: preAuthPaymentRequestCommon

Model Parameters

preAuthPaymentRequestCommon

All Of
paymentSession...

- judoId string *
- yourConsumerReference string *
- yourPaymentReference string *
- yourPaymentMetaData object
- currency string
- amount number<float> *
- webPaymentReference string
- clientDetails object
- initialRecurringPayment boolean
- primaryAccountDetails object

```
{
  "judoId": "",
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "yourPaymentMetaData": {},
  "currency": "",
  "amount": "",
  "webPaymentReference": "",
  "clientDetails": {
    "key": "",
    "value": ""
  },
  "initialRecurringPayment": false,
  "primaryAccountDetails": {
    "name": "",
    "accountNumber": "",
    "dateOfBirth": "",
    "postCode": ""
  }
}
```

24.8.40. Model: preAuthRequestApplePay

Model Parameters

- **preAuthRequestApplePay** — All Of: applePayRequest +
  - **pkPayment** `object` *
  - **cardAddress** `cardAddressRequestAttributes` — All Of: cardAddressCom...
  - **judoId** `string` *
  - **yourConsumerReference** `string` *
  - **yourPaymentReference** `string` *
  - **yourPaymentMetaData** `object` *
  - **currency** `string` *
  - **amount** `number<float>` *
  - **webPaymentReference** `string`
  - **clientDetails** `object`
  - **initialRecurringPayment** `boolean`
  - **primaryAccountDetails** `object`
  - **allowIncrement** `boolean`

```
{
  "pkPayment": {
    "token": {
      "paymentMethod": {
        "displayName": "",
        "network": "",
        "type": ""
      },
      "paymentData": {
        "data": "",
        "signature": "",
        "header": {
          "publicKeyHash": "",
          "ephemeralPublicKey": "",
          "transactionId": ""
        },
        "version": ""
      }
    }
  },
  "cardAddress": {},
  "judoId": "",
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "yourPaymentMetaData": {},
  "currency": "",
  "amount": "",
  "webPaymentReference": "",
  "clientDetails": {
    "key": "",
    "value": ""
  },
  "initialRecurringPayment": false,
  "primaryAccountDetails": {
    "name": "",
    "accountNumber": "",
    "dateOfBirth": "",
    "postCode": ""
  },
  "allowIncrement": false
}
```

24.8.41. Model: preAuthRequestCardToken

Model Parameters

- preAuthRequestCardToken — All Of: cardTokenPreAut...
  - cardToken string *
  - cv2 string
  - cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...
  - judoId string *
  - yourConsumerReference string *
  - yourPaymentReference string *
  - yourPaymentMetaData object *
  - currency string *
  - amount number<float> *
  - webPaymentReference string
  - clientDetails object
  - initialRecurringPayment boolean
  - primaryAccountDetails object
  - acceptHeaders string
  - userAgent string
  - cardHolderName string
  - emailAddress string
  - shippingAddress object
  - mobileNumber string
  - phoneCountryCode string

- **threeDSecure** `object`
- **threeDSecureMpi** `object`
- **recurringPayment** `boolean`
- **recurringPaymentType** `string<RECURRING | MI…`
- **relatedReceiptId** `string`
- **relatedPaymentNetworkTransactionId** `string`
- **allowIncrement** `boolean`
- **disableNetworkTokenisation** `boolean`

```
{
  "cardToken": "",
  "cv2": "",
  "cardAddress": {},
  "judoId": "",
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "yourPaymentMetaData": {},
  "currency": "",
  "amount": "",
  "webPaymentReference": ""
```

24.8.42. Model: preAuthRequestGooglePay

Model Parameters

- **preAuthRequestGooglePay** — All Of: googlePayReque...
  - **googlePayWallet** `object` *
  - **cardAddress** `cardAddressRequestAttributes` — All Of: cardAddressCom...
  - **judoId** `string` *
  - **yourConsumerReference** `string` *
  - **yourPaymentReference** `string` *
  - **yourPaymentMetaData** `object` *
  - **currency** `string` *
  - **amount** `number<float>` *
  - **webPaymentReference** `string`
  - **clientDetails** `object`
  - **initialRecurringPayment** `boolean`
  - **primaryAccountDetails** `object`
  - **acceptHeaders** `string`
  - **userAgent** `string`
  - **cardHolderName** `string`
  - **emailAddress** `string`
  - **shippingAddress** `object`
  - **mobileNumber** `string`
  - **phoneCountryCode** `string`
  - **threeDSecure** `object`

- **threeDSecureMpi** `object`
- **allowIncrement** `boolean`

```
{
  "googlePayWallet": {
    "token": "",
    "cardNetwork": "",
    "cardDetails": ""
  },
  "cardAddress": {},
  "judoId": "",
  "yourConsumerReference": ""
```

24.8.43. Model: preAuthRequestPan

Model Parameters

- preAuthRequestPan — All Of: cardDetailsCv2N...
  - cardNumber string *
  - cv2 string
  - expiryDate string *
  - startDate string
  - cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...
  - judoId string *
  - yourConsumerReference string *
  - yourPaymentReference string *
  - yourPaymentMetaData object *
  - currency string *
  - amount number<float> *
  - webPaymentReference string
  - clientDetails object
  - initialRecurringPayment boolean
  - primaryAccountDetails object
  - acceptHeaders string
  - userAgent string
  - cardHolderName string
  - emailAddress string
  - shippingAddress object

- **mobileNumber** `string`
- **phoneCountryCode** `string`
- **threeDSecure** `object`
- **threeDSecureMpi** `object`
- **recurringPayment** `boolean`
- **recurringPaymentType** `string<RECURRING | MI…`
- **relatedReceiptId** `string`
- **relatedPaymentNetworkTransactionId** `string`
- **allowIncrement** `boolean`
- **disableNetworkTokenisation** `boolean`

```
{
  "cardNumber": "",
  "cv2": "",
  "expiryDate": "",
  "startDate": "",
  "cardAddress": {},
  "judoId": "",
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "yourPaymentMetaData": {},
  "currency": "",
  "amount": "",
  "webPaymentReference": ""
```

24.8.44. Model: primaryAccountDetailsRequest

Model Parameters

- `primaryAccountDetailsRequest`
  - **primaryAccountDetails** `object`


```
{
  "primaryAccountDetails": {
    "name": "",
    "accountNumber": "",
    "dateOfBirth": "",
    "postCode": ""
  }
}
```

24.8.45. Model: recurringPaymentRequest

Model Parameters

- `recurringPaymentRequest`
  - **recurringPayment** `boolean`
  - **recurringPaymentType** `string<RECURRING | MI…`
  - **relatedReceiptId** `string`
  - **relatedPaymentNetworkTransactionId** `string`


```
{
  "recurringPayment": false,
  "recurringPaymentType": "",
  "relatedReceiptId": "",
  "relatedPaymentNetworkTransactionId": ""
}
```

24.8.46. Model: refundRequest

Model Parameters

refundRequest

- receiptId `string` *
- amount `number<float>`
- yourPaymentReference `string` *

```
{
  "receiptId": "",
  "amount": "",
  "yourPaymentReference": ""
}
```

24.8.47. Model: registerCardRequest

Model Parameters

registerCardRequest — All Of: object +

- yourConsumerReference string *
- yourPaymentReference string
- clientDetails object
- judoId string
- currency string
- initialRecurringPayment boolean
- cardHolderName string
- emailAddress string
- shippingAddress object
- mobileNumber string
- phoneCountryCode string
- threeDSecure object
- threeDSecureMpi object
- cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...
- primaryAccountDetails object

24.8.48. Model: checkCardRequest

Model Parameters

checkCardRequest — All Of: object +

- yourConsumerReference string *
- yourPaymentReference string
- clientDetails object
- judoId string
- currency string
- initialRecurringPayment boolean
- disableNetworkTokenisation boolean
- cardHolderName string
- emailAddress string
- shippingAddress object
- mobileNumber string
- phoneCountryCode string
- threeDSecure object
- threeDSecureMpi object
- cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...
- primaryAccountDetails object

```
{
  "yourConsumerReference": "",
  "yourPaymentReference": "",
  "clientDetails": {
    "key": "",
    "value": ""
  },
  "judoId": "",
  "currency": "",
  "initialRecurringPayment": false,
  "disableNetworkTokenisation": false,
  "cardHolderName": "",
  "emailAddress": "",
  "shippingAddress": {
    "address1": "",
    "address2": "",
    "town": "",
    "postCode": "",
    "countryCode": "",
    "isBillingAddress": false
  },
  "mobileNumber": "",
  "phoneCountryCode": "44",
  "threeDSecure": {
    "authenticationSource": "",
    "methodNotificationUrl": "https://api.judopay.com/order/3ds/methodNotification",
    "challengeNotificationUrl": "https://api.judopay.com/order/3ds/challengeNotification"
  },
  "threeDSecureMpi": {
    "dsTransId": "",
    "cavv": "",
    "eci": "",
    "threeDSecureVersion": ""
  },
  "cardAddress": {},
  "primaryAccountDetails": {
    "name": "",
    "accountNumber": "",
    "dateOfBirth": "",
    "postCode": ""
  }
}
```

24.8.49. Model: registerCardRequestPan

Model Parameters

registerCardRequestPan

All Of
cardDetailsCv2N...

- cardNumber string *
- cv2 string
- expiryDate string *
- startDate string
- yourConsumerReference string *
- yourPaymentReference string
- clientDetails object
- judoId string
- currency string
- initialRecurringPayment boolean
- cardHolderName string
- emailAddress string
- shippingAddress object
- mobileNumber string
- phoneCountryCode string
- threeDSecure object
- threeDSecureMpi object
- cardAddress cardAddressRequestAttributes

All Of
cardAddressCom...

- primaryAccountDetails object

```
{
  "recordNumber": ""
```

24.8.50. Model: resume3dsRequest

Model Parameters

resume3dsRequest

All Of
object +

- cv2 string
- threeDSecure object *
- primaryAccountDetails object

```
{
  "cv2": "",
  "threeDSecure": {
    "methodCompletion": ""
  },
  "primaryAccountDetails": {
    "name": "",
    "accountNumber": "",
    "dateOfBirth": "",
    "postCode": ""
  }
}
```

24.8.51. Model: riskParametersResponse

Model Parameters

- riskParametersResponse
  - **postCodeCheck** `string<PASSED | FAILED | U…`
  - **cv2Check** `string<PASSED | FAILED | NOT_PR…`
  - **merchantSuggestion** `string`


```
{
  "postCodeCheck": "",
  "cv2Check": "",
  "merchantSuggestion": ""
}
```

24.8.52. Model: networkTokenisationDetailsResponse

Model Parameters

- networkTokenisationDetailsResponse
  - networkTokenProvisioned boolean
  - networkTokenUsed boolean
  - virtualPan object


```
{
  "networkTokenProvisioned": false,
  "networkTokenUsed": false,
  "virtualPan": {
    "lastFour": "",
    "expiryDate": ""
  }
}
```

24.8.53. Model: saveCardRequestCommon

Model Parameters

saveCardRequestCommon — All Of: object +

- yourConsumerReference string *
- judoId string
- currency string
- cardHolderName string
- disableNetworkTokenisation boolean
- cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...


```
{
  "yourConsumerReference": "",
  "judoId": "",
  "currency": "",
  "cardHolderName": "",
  "disableNetworkTokenisation": false,
  "cardAddress": {}
}
```

24.8.54. Model: saveCardRequestPan

Model Parameters

saveCardRequestPan — All Of: object +

- cardNumber string *
- expiryDate string *
- startDate string
- yourConsumerReference string *
- judoId string
- currency string
- cardHolderName string
- disableNetworkTokenisation boolean
- cardAddress cardAddressRequestAttributes — All Of: cardAddressCom...

```
{
  "cardNumber": "",
  "expiryDate": "",
  "startDate": "",
  "yourConsumerReference": "",
  "judoId": "",
  "currency": "",
  "cardHolderName": "",
  "disableNetworkTokenisation": false,
  "cardAddress": {}
}
```

24.8.55. Model: scaExemptionRequest

Model Parameters

scaExemptionRequest

Example

transactionRiskAnalysis

```
{}
```

Indicates reason why challenge may not be necessary (this may be over-ruled by the issuer). Should not be specified in the same request as challengeRequestIndicator

24.8.56. Model: threeDSecureChallengeRequest

Model Parameters

- threeDSecureChallengeRequest
  - acceptHeaders string
  - userAgent string


```
{
  "acceptHeaders": "text/html",
  "userAgent": ""
}
```

24.8.57. Model: threeDSecureCompletedResponse

Model Parameters

- threeDSecureCompletedResponse
  - attempted boolean
  - result string
  - eci string
  - challengeRequestIndicator string<NoPreferen…
  - scaExemption string<TrustedBeneficiary | …


```
{
  "attempted": false,
  "result": "",
  "eci": "",
  "challengeRequestIndicator": "",
  "scaExemption": ""
}
```


[Conditional] Only returned for transactions using 3DS

24.8.58. Model: threeDSecureTwoChallengeRequiredResp

Model Parameters

- threeDSecureTwoChallengeRequiredResponse
  - challengeUrl string
  - cReq string
  - version string
  - receiptId string
  - result string
  - message string
  - md string


```
{
  "challengeUrl": "",
  "cReq": "",
  "version": "",
  "receiptId": "",
  "result": "",
  "message": "",
  "md": ""
}
```

24.8.59. Model: threeDSecureTwoDeviceDetailsRequiredR

Model Parameters

- threeDSecureTwoDeviceDetailsRequiredRespon…
  - methodUrl string
  - version string
  - receiptId string
  - result string
  - message string
  - md string


```
{
  "methodUrl": "",
  "version": "",
  "receiptId": "",
  "result": "",
  "message": "",
  "md": ""
}
```

24.8.60. Model: threeDSecureTwoCheckCardRequest

Model Parameters

- `threeDSecureTwoCheckCardRequest`
  - **cardHolderName** `string`
  - **emailAddress** `string`
  - **shippingAddress** `object`
  - **mobileNumber** `string`
  - **phoneCountryCode** `string`
  - **threeDSecure** `object`
  - **threeDSecureMpi** `object`

```
{
  "cardHolderName": "",
  "emailAddress": "",
  "shippingAddress": {
    "address1": "",
    "address2": "",
    "town": "",
    "postCode": "",
    "countryCode": "",
    "isBillingAddress": false
  },
  "mobileNumber": "",
  "phoneCountryCode": "44",
  "threeDSecure": {
    "authenticationSource": "",
    "methodNotificationUrl": "https://api.judopay.com/order/3ds/methodNotification",
    "challengeNotificationUrl": "https://api.judopay.com/order/3ds/challengeNotification"
  },
  "threeDSecureMpi": {
    "dsTransId": "",
    "cavv": "",
    "eci": "",
    "threeDSecureVersion": ""
  }
}
```

24.8.61. Model: threeDSecureTwoRequest

Model Parameters

- `threeDSecureTwoRequest`
  - **cardHolderName** `string`
  - **emailAddress** `string`
  - **shippingAddress** `object`
  - **mobileNumber** `string`
  - **phoneCountryCode** `string`
  - **threeDSecure** `object`
  - **threeDSecureMpi** `object`

```
{
  "cardHolderName": "",
  "emailAddress": "",
  "shippingAddress": {
    "address1": "",
    "address2": "",
    "town": "",
    "postCode": "",
    "countryCode": "",
    "isBillingAddress": false
  },
  "mobileNumber": "",
  "phoneCountryCode": "44",
  "threeDSecure": {
    "authenticationSource": "",
    "methodNotificationUrl": "https://api.judopay.com/order/3ds/methodNotification",
    "challengeNotificationUrl": "https://api.judopay.com/order/3ds/challengeNotification"
    "challengeRequestIndicator": "noPreference",
    "scaExemption": ""
  },
  "threeDSecureMpi": {
    "dsTransId": "",
    "cavv": "",
    "eci": "",
    "threeDSecureVersion": ""
  }
}
```

24.8.62. Model: transactionReceiptResponse

Model Parameters

- transactionReceiptResponse
  - receiptId string
  - originalReceiptId string
  - yourPaymentReference string
  - type string<Payment | Refund | PreAuth |…
  - createdAt string<date-time>
  - result string
  - message string
  - judoId integer
  - merchantName string
  - appearsOnStatementAs string
  - originalAmount string
  - amountCollected string
  - netAmount string
  - amount string
  - currency string
  - recurringPaymentType string
  - acquirerTransactionId string
  - externalBankResponseCode string
  - authCode string
  - walletType integer

- **riskScore** `integer`
- **paymentNetworkTransactionId** `string`
- **allowIncrement** `boolean`
- **isIncrementalAuth** `boolean`
- **emailAddress** `string`
- **cardDetails** `cardDetailsResponse`
- **billingAddress** `cardAddressResponseAttributes` All Of **cardAddressCom...**
- **consumer** `consumerResponse`
- **device** `deviceResponse`
- **yourPaymentMetaData** `object`
- **threeDSecure** `threeDSecureCompletedResponse`
- **risks** `riskParametersResponse`
- **disableNetworkTokenisation** `boolean`
- **networkTokenisationDetails** `networkTokenisati…`

```
{
  "receiptId": "",
  "originalReceiptId": "",
  "yourPaymentReference": "",
  "type": "",
  "createdAt": "",
  "result": "",
  "message": "",
  "judoId": 0,
  "merchantName": "",
  "appearsOnStatementAs": "",
  "originalAmount": "",
  "amountCollected": "",
  "netAmount": "",
  "amount": "",
  "currency": "",
  "recurringPaymentType": "",
  "acquirerTransactionId": "",
  "externalBankResponseCode": "",
  "authCode": "",
  "walletType": 0,
  "riskScore": 0,
  "paymentNetworkTransactionId": "",
  "allowIncrement": false,
  "isIncrementalAuth": false
```

24.8.63. Model: transactionReceiptHistoricResponse

Model Parameters

- transactionReceiptHistoricResponse
  - receiptId string
  - originalReceiptId string
  - yourPaymentReference string
  - type string<Payment | Refund | PreAuth |…
  - createdAt string<date-time>
  - result string
  - message string
  - judoId integer
  - merchantName string
  - appearsOnStatementAs string
  - originalAmount string
  - amountCollected string
  - netAmount string
  - amount string
  - currency string
  - recurringPaymentType string
  - acquirerTransactionId string
  - externalBankResponseCode string
  - authCode string
  - postCodeCheckResult string

- **walletType** `integer`
- **acquirer** `string`
- **webPaymentReference** `string`
- **noOfAuthAttempts** `integer`
- **paymentNetworkTransactionId** `string`
- **allowIncrement** `boolean`
- **isIncrementalAuth** `boolean`
- **disableNetworkTokenisation** `boolean`
- **cardDetails** `cardDetailsResponse`
- **billingAddress** `cardAddressResponseAttributes` All Of cardAddressCom...
- **consumer** `consumerResponse`
- **device** `deviceResponse`
- **yourPaymentMetaData** `object`
- **threeDSecure** `threeDSecureCompletedResponse`
- **risks** `riskParametersResponse`
- **networkTokenisationDetails** `networkTokenisati…`

```
{
  "receiptId": "",
  "originalReceiptId": "",
  "yourPaymentReference": "",
  "type": "",
  "createdAt": "",
  "result": "",
  "message": "",
  "judoId": 0,
  "merchantName": "",
  "appearsOnStatementAs": "",
  "originalAmount": "",
  "amountCollected": "",
  "netAmount": "",
  "amount": "",
  "currency": "",
  "recurringPaymentType": "",
  "acquirerTransactionId": "",
  "externalBankResponseCode": "",
  "authCode": "",
  "postCodeCheckResult": "",
  "walletType": 0,
  "acquirer": "",
  "webPaymentReference": "",
  "noOfAuthAttempts": 0,
  "paymentNetworkTransactionId": "",
  "allowIncrement": false
```

24.8.64. Model: transactionReceiptListResponse

Model Parameters

- transactionReceiptListResponse
  - receiptId string
  - originalReceiptId string
  - yourPaymentReference string
  - type string<Payment | Refund | PreAuth |…
  - createdAt string<date-time>
  - result string
  - message string
  - judoId integer
  - appearsOnStatementAs string
  - originalAmount string
  - amountCollected string
  - netAmount string
  - amount string
  - currency string
  - postCodeCheckResult string
  - walletType integer
  - allowIncrement boolean
  - isIncrementalAuth boolean
  - cardDetails cardDetailsListResponse
  - consumer consumerResponse

- **yourPaymentMetaData** `object`
- **threeDSecure** `threeDSecureCompletedResponse`


```
{
  "receiptId": "",
  "originalReceiptId": "",
  "yourPaymentReference": "",
  "type": "",
  "createdAt": "",
  "result": "",
  "message": "",
  "judoId": 0,
  "appearsOnStatementAs": "",
  "originalAmount": "",
  "amountCollected": "",
  "netAmount": "",
  "amount": "",
  "currency": "",
  "postCodeCheckResult": "",
  "walletType": 0,
  "allowIncrement": false,
  "isIncrementalAuth": false,
  "cardDetails": {
    "cardLastfour": "",
    "endDate": "",
    "cardType": 0
  },
  "consumer": {
    "yourConsumerReference": ""
  },
  "yourPaymentMetaData": {},
  "threeDSecure": {
    "attempted": false,
    "result": "",
    "eci": "",
    "challengeRequestIndicator": "",
    "scaExemption": ""
  }
}
```

24.8.65. Model: voidRequest

Model Parameters

- voidRequest
  - receiptId string *
  - amount number<float>
  - yourPaymentReference string

```
{
  "receiptId": "",
  "amount": "",
  "yourPaymentReference": ""
}
```