Plunet BusinessManager 8.5

Plunet API: Release Notes

AUTHOR

Plunet GmbH

Plunet

Contents

Introduction

This document provides an overview of all the changes that have been made to the Plunet API since version 7.4.

Version 8.5

New features

Access to the project management folder via DataDocument30

Using the API methods of the *DataDocument30* web service it is now also possible to query the project management folders of quotes and orders. Please refer to the *FolderTypes* Enum of API Javadoc.

> NOTE
> This call requires the *Project team* permission for quotes/orders. Read-only permissions are not evaluated.

Changes made to existing calls

Time zone of contact person is now taken into account in requests

When a contact person is set in a request using *DataRequest30.setCustomerContactID()*, the time zone of the contact person is now also applied if it differs from the customer itself.

> NOTE
> Please always call .setCustomerID first, before you set the contact person.

Version 8.4

New features

Access to the order date

In previous versions of Plunet Business Manager the calls *DataOrder30.get/setCreationDate()* raised the expectation that it can be used to access the order date. This is not correct because the creation date is an internal field that is only visible in the database. Furthermore, the setter was not working at all.

We have now added some new calls:

- *DataOrder30.getOrderDate()*: The call returns the order date, which is the field that is also visible in the UI.
- *DataOrder30.setOrderDate()*: The call allows to modify the order date.

Deprecations

DataOrder30.setCreationDate(): The call was not working and in general the creation timestamp will not be modified at all, so the call is now deprecated. Please remove from your implementation as soon as possible.

Backwards compatibility changes

Changes to the Order/OrderIN object

Due to the former changes we also modified the definition of the order objects. The *creationDate* has been removed and instead the *orderDate* has been added. Please recreate the stubs for your implementation in case you have issues.

Version 8.3

New features

Setting a resource in a job

As per our announcement *DataJob30.setResourceID()* has been disabled. Plunet 8.3 now provides a replacement for this call:

- *DataJobRound30.setResourceForReview()*: This call will be an equivalent to the *Review* button in the new *Assignment* tab when doing manual assignment. In addition to the *roundID* it takes a *resourceID* and an optional *resourceContactID* as parameters. If you do not want to set a specific contact person, please set *resourceContactID* to 0.

Assigning a job in review

If a job already has a resource selected for review (either via API or UI) a simplified call to assign the job has been added:

- *DataJobRound30.assignResource_InReview()*: This call will represent a click on the *Assign* button after the resource has been preselected. It only takes the *roundID* as a parameter.

Assigning a job to a resource

For jobs with the status *In preparation* a manual assignment can now be made via API using the new call:

- *DataJobRound30.assignResource()*: According to the call *.setResourceForReview()* this call takes the *resourceID* and the optional *resourceContactID* as parameters. The assignment will be executed instantly.

Version 8.2

New features

Getting all available resources for a job

There is a new call for the *DataJobRound30* web service:

DataJob30Round.getResourcesForRound(): The call will return a list of all available resources for a round based on selection criteria, availability and (Plunet) permissions. Based on cascading job round settings it will auto-filter for resources that should no longer be used due to the result of the previous round.

Version 8.1

New features

Improved handling of project templates

The API calls *DataQuote30/DataOrder30.getTemplateList()* return an object that now also includes the template name – not only the description as it used to be.

Version 8.0

Highlights

Support for item callbacks

It is now possible to work with callbacks on item scope. This especially allows you to implement complex workflows with partial project delivery or interactions within *Multi-team projects*.

Query existing callbacks

We implemented a call that enables all registered callbacks to be fetched for an API User. An administrator can also query all existing callbacks in the system.

Job display number

You can now receive the serial number of a job within an item. This allows you to map remote information to a job in Plunet.

Search for payables

There is a new call that implements a reporting feature for payables. This can be used for your custom integrations to accounting tools, for example.

Backwards compatibility changes

Support for setting resources in jobs has been removed

The new advanced assignment concept for jobs introduced with Plunet 8.0 no longer allows a resource to be directly set in a job. The corresponding API calls are also disabled.

DataJob30.assignJob() and *.setResourceID()* are therefore disabled and the *resourceID* property of the *JobIN* object will no longer be evaluated for *.insert3()* and *.update* call.

Replacements for these calls were added in Plunet 8.3. Please refer to the release information for Plunet 8.3 in this document.

Deprecations

- *DataOrder30.getDocuments_Within_FinalFolder()*: This call is deprecated, please use *DataDocument30.getFileList* with folderType = 12 instead.
- *DataOrder30.getDocuments_Within_ReferenceFolder_ByLanguage()*: This call is deprecated, please use *DataDocument30.getFileList* with folderType = 5 instead.

- *DataOrder30.getDocuments_Within_ReferenceFolder()*: This call is deprecated, please use *DataDocument30.getFileList* with folderType = 5 instead.
- *DataOrder30.getDocuments_Within_SourceFolder_ByLanguage()*: This call is deprecated, please use *DataDocument30.getFileList* with folderType = 6 instead.
- *DataOrder30.getDocuments_Within_SourceFolder()*: This call is deprecated, please use *DataDocument30.getFileList* with folderType = 6 instead.

New features

Below you can find the changes made to the Plunet API since the release of Plunet 7.4. The changes are grouped into new, reworked and deprecated features.

Job rounds

Within the changes made to the resource assignment, the way of assigning a resource via Plunet API has changed as well. For this purpose we provide a whole new web service: *DataJobRound30*. This endpoint provides all necessary tools for the assignment process.

Please note that this endpoint currently only supports order jobs and quote jobs.

New calls

- *DataJobRound30.addRound()*: Adds a new round to the job. Returns the roundID.
- *DataJobRound30.getRoundObject()*: Returns the round (if it exists).
- *DataJobRound30.updateRound()*: Updates an existing round by input object.
- *DataJobRound30.delete()*: Deletes an existing round based on its roundID.
- *DataJobRound30.getAssignedRoundID()*: Returns the assigned round of the job (if it exists).
- *DataJobRound30.getAllRoundIDs()*: Returns the *roundIDs* of all rounds of the job.
- *DataJobRound30.getSearchCriteria()*: Returns the selection criteria of the round.
- *DataJobRound30.setSearchCriteria()*: Sets the selection criteria of the round.

- *DataJobRound30.getRankingMethods()*: Returns the ranking criteria of the round.
- *DataJobRound30.setRankingMethods()*: Sets the ranking criteria of the round.

Other new features

API Superlock User

It is now possible to create one API Superlock User per Plunet BusinessManager instance. This user always has priority when it comes to locking. For example, if an order is locked by a UI user, the Superlock User can revoke the lock and overwrite the locked order. These superlocking rights can be assigned under Admin → Users just like for regular Plunet API users.

> NOTE
> This feature is not available by default. Please contact Plunet Support if you would like to use it. Please note that you should consider implementing a queue instead, in order to not override actions being done in the UI.

Company codes for invoices

- *DataAdmin30.getCompanyCodeList()*: Returns a list of all company codes available in your Plunet BusinessManager.
- *DataOutgoingInvoice30.getCompanyCode()*: Returns the company code of the invoice.
- *DataCreditNote30.getCompanyCode()*: Returns the company code of the credit note.
- *DataPayable30.getCompanyCode()*: Returns the company code of the payable.

Search for payables via Plunet API

- *DataPayable30.search()*: You can now search for payables via API. The search criteria are specified by the *SearchFilter_Payable*.
- *SearchFilter_Payable*: Search filter for the payable search. It is possible to filter by resource, export flag, status, a time frame, currency and a list of text modules.

Querying existing callbacks

The new call *DataAdmin30.getListOfRegisteredCallbacks()* allows API users to read all registered remote end points for this user. An API user with administrator privileges will receive all registered callback entries for any API user.

Item callbacks

Support for callbacks on item level has been added.

- *DataItem30.registerCallback_Notify()/.deregisterCallback_Notify()*: Registers/deregisters a listener for a specific event across all items in the system.
- *DataItem30.registerCallback_Observer().deregisterCallback_Observer()*: Registers/deregisters a listener for any events triggered on a specific item.

External SAML ID

- *DataResourceContact30.getSamlExternalID()*: Returns the field samlExternalID of the resource contact person.
- *DataResourceContact30.setSamlExternalID()*: Sets the field samlExternalID of the resource contact person.
- *DataResource30.getSamlExternalID()*: Returns the field samlExternalID of the resource.
- *DataResource30.setSamlExternalID()*: Sets the field external SAML ID of the resource.

Various calls

- *DataItem30.setDeliveryDate()*: Set the delivery date of an item. Only available for order items.
- *DataItem30.getDeliveryDate()*: Get the delivery date of an item. Only available for order items.
- *DataItem30.getDeliveryDate()*: Returns the delivery date of the job.
- *DataAdmin30.getSystemCurrencies()*: Returns a list of all currencies available.
- *DataJob30.getCurrency()*: Returns the job currency represented by its ISO code.

- *DataJob30.getJobNumber()*: Returns the job number (e.g. “001”).
- *DataPayable30.getPayableID()*: Returns the payable ID corresponding to the item.
- *DataJob30.getPayableID()*: Returns the payable ID corresponding to the job.
- *DataOrder30.createLink()*: Creates a link between two orders.

Improved features

- New entry *HYPER_LINK(7)* for enum *TextModuleType*.
- The support for *FolderType INVOICE(13)* has been added to the calls of *DataDocument30*.
- *DataOrder30.insert2()*: If no project manager is set, the preselected project manager will be set automatically.

Change log 7.4

Below you can find the changes made to the Plunet API for Plunet version 7.4. The changes are grouped into new, reworked and deprecated features.

Security issue fixes

Within the upgrade to Plunet 7.4, multiple potential security issues have been fixed:

UNC paths deactivated

A couple of calls for copying files used an UNC paths as input parameter. This is a potential security issue, because an attacker may submit paths of untrusted UNC locations. Furthermore, Plunet cannot differentiate between user folder access rights – a customer API user may access any folders that Plunet BusinessManager can access.

The following calls are affected:

- *DataRequest25/30.copyDocument_toReferenceFolder*: This call is no longer supported. Please use *DataDocument30* with *FolderType* 1 or 2.

- *DataRequest25/30.copyDocument_toSourceFolder*: This call is no longer supported. Please use *DataDocument30* with *FolderType* 1 or 2.
- *DataItem25/30.setCatReport*: This call is no longer supported. Please use the new call *DataItem25/30.setCatReport2* that works with byte streams instead of UNC paths.
- *DataJob25/30.setCatReport*: This call is no longer supported. Please use the new call *DataJob25/30.setCatReport2* that works with byte streams instead of UNC paths.

For customers who cannot adjust their implementation, please contact our support to reactivate the old calls.

String sanitizing to prevent XSS attacks

Various calls allowed string inputs containing potential malicious JavaScript code. With Plunet BusinessManager 7.4, all strings submitted via API are sanitized to ensure XSS attacks are no longer possible.

This update has also been applied to Plunet BM 7.3.

New features

Using the two new API services DataPayable30, DataCreditNote30 and the existing service DataOutgoingInvoice30, incoming and outgoing invoices can now be fully managed via the Plunet API. The services allow the creation, reading and modification of invoices in all details, including items and price lines. Plunet thus enables its users to automate central components of project management by connecting financial accounting software.

> NOTE
> Due to legal restrictions, customers using the SAF-T PT integration are limited to read-only calls regarding invoices, payables and credit notes.

Payables

- *DataPayable30.get/setIsExported*: Get or set the export flag.
- *DataPayable30.get/setStatus*: Get or set the *PayableStatus*.

- *DataPayable30.get/setInvoiceDate*: Get or set the invoice date.
- *DataPayable30.get/setValueDate*: Get or set the value date.
- *DataPayable30.get/setPaymentDueDate*: Get or set the payment due date.
- *DataPayable30.get/setPaidDate*: Get or set the date the payable is paid.
- *DataPayable30.getExpenseAccount*: Get the invoice expense account.
- *DataPayable30.get/setCreditorAccount*: Get or set the creditor account.
- *DataPayable30.getTotalNetAmount*: Get the total net amount.
- *DataPayable30.getTotalTaxAmount*: Get the total tax amount.
- *DataPayable30.getInvoiceTaxTypes*: Get a list containing the invoice tax types used within the payable.
- *DataPayable30.get/setMemo*: Get or set the memo.
- *DataPayable30.getResourceID*: Get the resource ID.
- *DataPayable30.getPaymentCreatorResourceID*: Get the resource that created the payment.
- *DataPayable30.get/setExternalInvoiceNumber*: Get or set the external invoice number.
- *DataPayable30.get/setAccountStatement*: Get or set the account statement.
- *DataPayable30.getPaymentMethod*: Get the payment method.
- *DataPayable30.getCurrency*: Get the currency of the payable.

Furthermore, it is possible to create, read, update and delete items:

- *DataPayable30.insertPayableItem*: Insert a new payable item.
- *DataPayable30.updatePayableItem*: Update an existing payable item.
- *DataPayable30.deletePayableItem*: Delete a payable item.
- *DataPayable30.getPayableItemList*: Return a list containing all items related to the payable.

Price lines are also supported for payables:

- *DataPayable30.insertPriceLine*: Insert a new price line into the payable item.
- *DataPayable30.updatePriceLine*: Update an existing price line.
- *DataPayable30.deletePriceLine*: Delete a price line.
- *DataPayable30.getPriceLine_List*: Get all price lines for a payable item.
- *DataPayable30.getPriceUnit_List*: Return a list of all price units available for a given service.
- *DataPayable30.getPriceUnit*: Get the price unit object.
- *DataPayable30.getServices_List*: Get a list of all services available.

Receivables

A couple of new calls have been added to the webservice *DataOutgoingInvoice30*:

- *DataOutgoingInvoice30.getTaxByTypeAndCurrencyType*: Get the tax amount of the outgoing invoice filtered by tax type. The value is calculated in the project (default) currency or domestic currency.
- *DataOutgoingInvoice30.getInvoiceTaxTypes*: Return a list of all tax types used within the invoice.
- *DataOutgoingInvoice30.get/setContactPersonID*: Get or set the contact person.
- *DataOutgoingInvoice30.get/setAddressID*: Get or set the address.
- *DataOutgoingInvoice30.get/setPaidDate*: Get or set the paid date.
- *DataOutgoingInvoice30.get/setPaymentDueDate*: Get or set the payment due date.
- *DataOutgoingInvoice30.get/setIsInvoiceExported*: Get or set the export flag of the invoice.
- *DataOutgoingInvoice30.get/setPONumber*: Get or set the PO number.
- *DataOutgoingInvoice30.createInvoiceDocument*: Create an invoice document as an RTF file. Returns the UNC path of the file.
- *DataOutgoingInvoice30.getInvoiceDocuments*: Return the relative network paths for all available invoice documents.

Credit notes

With the release of Plunet 7.4, it is possible to read and modify credit notes. The new endpoint *DataCreditNote30* offers a wide variety of calls:

- *DataCreditNote30.getCreditNoteNr*: Get the credit note number.
- *DataCreditNote30.get/setCustomerID*: Get or set the customer.
- *DataCreditNote30.getInvoiceID*: Get the invoice ID related to the credit note.
- *DataCreditNote30.get/setStatus*: Get or set the *CreditNoteStatus*.
- *DataCreditNote30.getNet/getNetByCurrencyType*: Get the net amount of the credit note. The value is calculated in the project (default) currency or domestic currency.
- *DataCreditNote30.getGross/getGrossByCurrencyType*: Get the gross amount of the credit note. The value is calculated in the project (default) currency or domestic currency.
- *DataCreditNote30.getOutstanding/getOutstandingByCurrencyType*: Get the outstanding amount of the credit note. The value is calculated in the project (default) currency or domestic currency.
- *DataCreditNote30.getTax/getTaxByCurrencyType /getTaxByTypeAndCurrencyType*: Get the tax amount of the credit note. The value is calculated in the project (default) currency or domestic currency and can be filtered by tax type.
- *DataCreditNote30.getTaxTypes*: Return a list of all tax types used in the credit note.
- *DataCreditNote30.get/setCreditDate*: Get or set the credit date.
- *DataCreditNote30.get/setReceivableAccount*: Get or set the receivable account.
- *DataCreditNote30.get/setRevenueAccount*: Get or set the revenue account.
- *DataCreditNote30.get/setBriefDescription*: Get or set the brief description.
- *DataCreditNote30.get/setSubject*: Get or set the subject.
- *DataCreditNote30.get/setContactPersonID*: Get or set the contact person.
- *DataCreditNote30.get/setAdressID*: Get or set the address.
- *DataCreditNote30.get/setPaidDate*: Get or set the paid date.

- *DataCreditNote30.get/setIsExported*: Get or set the export flag.
- *DataCreditNote30.get/setPONumber*: Get or set the PO number.

It is possible to create, read, update and delete items:

- *DataCreditNote30.insertCreditNoteItem*: Insert a new credit note item.
- *DataCreditNote30.updateCreditNoteItem*: Update an existing credit note item.
- *DataCreditNote30.deleteCreditNoteItem*: Delete a credit note item.
- *DataCreditNote30.getCreditNoteItemList*: Return a list containing all payment items related to the credit note.

Price lines are supported for credit notes:

- *DataCreditNote30.insertPriceLine*: Insert a new price line into a credit note item.
- *DataCreditNote30.updatePriceLine*: Update an existing price line.
- *DataCreditNote30.deletePriceLine*: Delete a price line.
- *DataCreditNote30.getPriceLine*: Get an existing *PriceLine* object.

Add workflow jobs to an item

You can select a workflow and add all of its jobs to an item. The job will only be copied to the item if the language combinations are compatible.

- *DataItem30.copyJobsFromWorkflow*: Copy the jobs in the workflow to the item.
- *DataCustomer30.getAvailableWorkflows*: Get all the workflows available for a specific customer.
- *Workflow*: Object consisting of a workflow ID, name, description, workflow status and workflow type.
- *WorkflowStatus*: Enum defining the status of a workflow. Contains INPREPARATION(0), RELEASED(1), CANCELED(2) and RELEASED_FOR_SELECTION(3).

- *WorkflowType*: Enum defining the type of a workflow. Contains STANDARD (0), ORDER (1) and QUOTE_ORDER (2).

Furthermore, you can preselect a workflow for a given request:

- *DataRequest30.get/setWorkflowID*: Get or set the workflow for a request.

Request order/quote

There are a few new ways of creating orders and quotes based on existing requests:

- *DataQuote30.requestOrder*: Create a new order based on the request values. This works in a similar way to the “Place order” button in the customer portal.
- *DataQuote30.requestQuote*: Create a new quote based on the request values. This works in a similar way to the “Request a quote” button in the customer portal.
- *DataRequest30.quoteRequest*: Turn a request into a quote. This works in a similar way to the existing *DataRequest30.orderRequest*.

Language-independent items

You can manipulate language-independent items via Plunet API by using the following calls:

- *DataItem30.insertLanguageIndependentItem*: Insert a language-independent item via ItemIN object. Only one language-independent item is allowed for each project.
- *DataItem30.getLanguageIndependentItemObject*: Get the language-independent item from a project, if it exists.

Various calls

- *DataQuote30.get/setCustomerContactID*: Set the customer contact for a quote.
- *DataJob30.get/setResourceContactPersonID*: Get or set the resource contact person of a job (specified by its contact person ID).
- *DataQuote30.get/setProjectCategory*: Get or set the project category of a quote.

- *DataOrder30.get/setProjectCategory*: Get or set the project category of an order.
- *DataJob30.get/setDescription*: Get or set the description of a job.
- *DataJob30.get/setComment*: Get or set the comment of a job.
- *DataItem25/30.setCatReport2*: Set the CAT report. Requires a byte stream representation of the CAT report as input parameter.

Reworked and improved features

- Notify callbacks: When registering an existing notify callback, the entry will be overwritten with the new remote address.
- *DataJob30.get/setDueDate*: Fixed a bug where the delivery would also be set.
- When working with *SearchFilter* objects, the parameter languageCode will be set to “EN” by default.
- When searching for quotes via API, it is now possible to filter by customer.
- *DataCustomerContact30.get/setStatus*: Added the enum *ContactPersonStatus* that depicts the different status.
- *DataOutgoingInvoice30.setRevenueAccount*: Changed the *accountID* parameter type from string to int.
- When creating or updating a job price line via Plunet API, the *Memo* field is ignored.
- *DataResource30.setPaymentInfo*: Changed the parameter name from *customerID* to *resourceID*.
- When inserting a new object using the *DataXXX25* webservices, the object is now fetched automatically.
- When inserting a new object using the *DataXXX25* webservices, the object is now fetched automatically.
- *DataCustomerAddress30.getAddressType*: Renamed parameter from *customerID* to *addressID*.
- *DataOutgoingInvoice30.setValueDate*: Now correctly sets the field *Value date* instead of the field *Invoice date*.
- Downloads of documents via Plunet API are now logged in the change log.

- Only one session per user is allowed. If you log in twice with the same user, the previous session is invalidated.

Deprecated features

We do not recommend using the deprecated features anymore, as an improved version has been implemented. However, we will ensure that the functionality is still provided. If you are already working with these calls, you do not need to change your implementation.

- *DataJob30.setPriceListeID*: Please use *DataJob30.setPriceList* instead.

The following calls were removed by default from Plunet 7.4. If you still need to work with them, please contact our support team.

- *DataRequest25/30.copyDocument_toReferenceFolder/copyDocument_toSouceFolder*: Please use the calls provided by *DataDocument30*.
- *DataItem25/30.setCatReport*: Please use *DataItem25/30.setCatReport2* instead.
- *DataJob25/30.setCatReport*: Please use *DataJob25/30.setCatReport2* instead.

The following calls were removed from Plunet 7.4.

- *DataOutgoingInvoice/DataOutgoingInvoice25.setRevenueAccount*: No longer supported. Please use *DataOutgoingInvoice30.setRevenueAccount* instead.

Copyright Notice

Plunet GmbH
Dresdener Str. 15
10999 Berlin