



# **Cúram 8.1.3**

## **Financials Guide**



## Note

---

Before using this information and the product it supports, read the information in [Notices on page 75](#)



# Edition

---

This edition applies to Cúram 8.1, 8.1.1, 8.1.2, and 8.1.3.

© Merative US L.P. 2012, 2024

Merative and the Merative Logo are trademarks of Merative US L.P. in the United States and other countries.



# Contents

---

<b>Note.....</b>	<b>iii</b>
<b>Edition.....</b>	<b>v</b>
<b>1 Cúram Financials guide.....</b>	<b>9</b>
1.1 Understanding Cúram financials.....	9
Payments, liabilities, and payments received.....	9
Financial components.....	10
Case nominee and preferred delivery pattern.....	11
Financial instruction line items and financial instructions.....	13
Financial instruments and payslips.....	16
1.2 Generating payments and liabilities.....	16
Simulating payments.....	17
Creating and expiring financial components.....	18
Generating financial instruction line items.....	18
Generating financial instructions.....	19
Generating financial instruments and payslips.....	20
Issuing payments online and to third parties.....	20
Examples of generating payments.....	21
1.3 Processing deductions and adjustments.....	23
Deductions.....	23
Tax adjustments.....	25
Surcharge adjustments.....	26
Financial account adjustments.....	26
Examples of payment generation with deductions and tax adjustments.....	27
Examples of liability with surcharge adjustments.....	30
1.4 Maintaining payments and liabilities.....	32
Capturing manual payments.....	33
Canceling and reissuing payments.....	33
Invalidating payments.....	34
Approving suspended payments.....	35
Reversing liabilities.....	35
Writing-off liabilities.....	36
Over and under payment processing.....	36
1.5 Processing and maintaining payments received.....	37
Recording payments received.....	38
Transferring payments received from a suspense account.....	38
Allocating payments received.....	38
Liability over allocation.....	39
Refunding an unallocated amount.....	39

Canceling a refund.....	40
Canceling payments received.....	41
1.6 Financial instruction types.....	41
Payment instruction.....	41
Liability instruction.....	43
Payment received instruction.....	44
Reversal instruction.....	45
Write-off instruction.....	47
Third-party payment instruction.....	47
Adjustment instruction.....	48
1.7 Financial batch processes.....	48
Submitting to the batch queue.....	48
Running the batch launcher.....	49
Running a batch program from the command line.....	49
Financials batch suite.....	49
Business processing date.....	50
Output to logs, emails, and reports.....	50
GenerateInstructionLineItems.....	51
GenerateInstruments.....	53
GeneratePayslips.....	54
LoadPaymentsReceived.....	54
IssueConcernPayments.....	55
ExpirePayments.....	56
ProcessPaymentInstrumentTypes.....	57
Payment reconciliation.....	58
GeneralLedgerInterface.....	58
ReconcileCaseAccount.....	60
1.8 Financial code tables.....	61
Financial code tables.....	61
Instruction Line Items (ILI) relationship types.....	63
1.9 Financial customization points.....	64
Financial hooks.....	65
Assessment customization points.....	71
<b>Notices.....</b>	<b>75</b>
Privacy policy.....	76
Trademarks.....	76

# 1 Cúram Financials guide

---

Cúram financials are used to process payments and liabilities for persons and employers. A nominee is required for financials to be generated. Payments can be captured, invalidated, canceled, or reissued. Liabilities can be reversed and written off.

**Note:** You can integrate the application financials with an Enterprise Resource Planning (ERP) financial system. For more information about the processing that occurs in the ERP financial system, see the *Cúram Financial Adapter Technical Overview Guide*. For more information about how to configure the application's financials with the ERP system, see the *Cúram Integrated ERP Operations Guide*. Payments and bills are issued in relation to eligible cases. For more information about case processing, see the *Cúram Integrated Case Management Guide*. Cúram deductions are a means of budgeting or clearing an existing debt from a benefit payment. For more information about Cúram deductions, see the *Cúram Deductions Guide*.

## 1.1 Understanding Cúram financials

---

Financial components, instruction line items, instructions, and instruments are the building blocks of Cúram financials. Cúram uses the building blocks to process outgoing payments and liabilities to receive payments into the system, and to complete other account maintenance tasks.

### Payments, liabilities, and payments received

Payments, liabilities, and payments received are the basic financial units.

- **Payments**

A payment is an issue of funds from the organization to a participant. Payments are issued to participants that are eligible for a benefit product. For example, a payment might be a \$300 check that is issued to a person who is eligible for income support payments. Payments are also generated for participants when the organization either underpays participants or over-bills participants. In the preceding example, if the organization was meant to pay the participant \$325, an under payment is created to pay the participant the extra \$25 that is due.

- **Liabilities**

A liability is a charge for funds from the organization to a participant. Liabilities are issued to participants that are eligible for a liability product. For example, a liability might be an invoice for \$100 issued to an employer who is liable for service fees. Liabilities are also generated for participants when the organization overpaid the participant or under-billed the participant. For example, in this example if the organization is meant to bill the employer for \$125, an under billing is created to bill the employer the extra \$25 that is due.

- **Payments received**

A payment received is an amount of money that is received by the organization and recorded on the system. Payments received are usually sent to the organization in response to a bill, that is, payments are used to pay off liabilities. Unlike payments and liabilities, payments received

are not generated as part of case processing. Instead, payments received are recorded on the system either manually by a user or by using a batch process that records payments that are received in bulk, for example all payments received by electronic fund transfer (EFT) from a particular bank. Payments received are associated with a person's financials through the allocation of the payment received toward one or more outstanding liabilities.

## Financial components

Financial components are schedules to pay or bill in relation to a case. Case components are the benefits or liabilities that a primary client might be eligible, for example an income support benefit.

A financial component is generated from a case component if the primary client is determined to be eligible for the case component.

The following table describes the information that is contained in each financial component.

*Table 1: The information that is contained in each financial component.*

Financial component	Explanation
Category.	Determines whether the financial component is a payment for a benefit or a bill for a liability.
Case nominee.	Specifies the participant to receive the payment or bill.
Primary client.	Specifies the client for whom entitlement, that is the amount to pay or bill, was calculated.
Amount.	Specifies the amount to pay or bill.
Delivery pattern information.	The following list outlines the two types of information that the delivery pattern information can specify: <ul style="list-style-type: none"> <li>• The delivery method and frequency, for example by cash on a weekly basis, by check on a monthly basis, and so on.</li> <li>• The cover period type and cover period offset, for example issue N number of days in advance where N is the cover period offset.</li> </ul>
Start date.	Specifies the start date for the payment or billing schedule.
End date.	Specifies the end date for the payment or billing schedule.
Scheduled due date.	Specifies the scheduled due date for the payment or bill.

### Sample financial component

A sample financial component might indicate that John Smith is to receive \$25 per week in payments by check on a Friday from 1 January through 1 April. The payment schedule might include a cover period of issue three days in advance. Each weekly payment has a due date on a Tuesday before the Friday, allowing for three days for the check to clear by Friday.

## Reassessment processing

Financial components are also created as part of reassessment processing, that is, financial components are created for any overpayment and underpayments that are issued on a case. For example, if John Smith is originally paid \$25, but a change in evidence makes him eligible for \$40, a financial component with an amount of \$15 is created to rectify the underpayment. For more information about overpayment and underpayment processing, see the *Over and underpayment processing* related link.

## Once-off and recurring financial components

Financial components are either once-off or recurring.

Once-off financial components are processed only one time. An example of a once-off financial component is an underpayment that is processed completely on a single date.

Recurring financial components are processed repeatedly according to their delivery frequency. Each time a recurring financial component is processed, the next due date is calculated by adding the nominee's delivery frequency to the current date. The next due date is rolled forward until the financial component reaches its end date.

Typically, recurring financial components are processed multiple times.

**Note:** Recurring financial components are processed only one time if their lifespan is shorter than or equal to the length of their delivery frequency.

## Secondary financial components

Secondary financial components are schedules to make deductions from payments. Secondary financial components are created when deductions are set up for benefit cases. Secondary financial components are processed when case payments are issued. A total deductible amount is calculated for each nominee who is receiving a payment on the case. The total deductible amount is the total amount that can be deducted from a payment that is issued to the nominee. During deduction processing, each deduction is processed against the reducing balance of the total deductible amount that is calculated for the nominee.

## Related concepts

[Over and under payment processing on page 36](#)

Case reassessment checks if changes in case circumstances may have resulted in a nominee being over or under paid.

## Case nominee and preferred delivery pattern

Financial components are issued to the nominee assigned to a case component and delivered according to the nominee's preferred delivery pattern.

- **Case nominee**

The default case nominee for all case components is the primary client. However, a case nominee can consist of any individual or party that is designated to receive a case component on behalf of the primary client.

- **Preferred delivery patterns**

The nominee that is assigned to a case component can indicate a preferred delivery pattern. A delivery pattern defines the frequency and method by which payments or bills are issued. For example, a delivery pattern might be weekly by check on Mondays or daily by invoice. If a delivery method for a delivery pattern is check, the case nominee that is assigned to receive the check must have an address that is recorded in the system. Similarly, if the delivery method is electronic funds transfer (EFT), the case nominee must have a bank account that is recorded in the system.

**Note:** Any currency with an active currency exchange rate can be assigned to a case nominee at the case level. The assigned currency is used for all case payments or bills for case components to which the case nominee is assigned. A case nominee's currency can be changed over the lifecycle of the case.

### ***Cover period patterns and offsets***

A delivery pattern also defines the cover period pattern for payment or bill delivery.

- **Cover period pattern**

A cover period pattern specifies how payments or bills are issued, for example in advance, in arrears, once-off, and so on. For example, the delivery pattern 'weekly by check on Mondays' with a cover period pattern of 'in advance' indicates that each payment occurs on a Monday and covers the week that starts on the Monday and continues to the next Sunday.

- **Offsets**

An offset typically defines the number of days in advance that a payment or bill must be processed to reach a case nominee on time. For example, check payments might be processed three days in advance so that the nominee receives the check on time. During financial component processing, the offset is used to calculate the due date of the financial component. There are two types of offsets: the delivery method offset and the cover pattern offset. The delivery method offset allows for the time it takes to process and print certain delivery methods. For example, it might take the organization two days to process and print check payments.

The cover period offset allows for the time it takes to deliver payments or bills by a specific delivery pattern. The cover pattern offset can be specified for a delivery pattern when the selected cover pattern includes an offset. For example, issue in advance N days before issue where N is the cover period offset that must be specified. For example, a two-day offset might be required for the weekly by check delivery pattern. The offset allows for the time it takes for a check to be sent through the mail.

If both offsets are used, the offsets are added during the financial component processing to calculate the due date. For example, if the delivery method, check, has an offset of two days and the delivery pattern, weekly by check, also has an offset of two days, then payments are due four days in advance.

**Note:** Offsets are set at the product level as part of financial administration. Offsets can also be modified as part of case creation.

### **Payment exclusion dates for delivery methods**

Payment exclusion dates represent the days on which the organization cannot make payments that use a particular delivery method. A prepayment requirement specifies that financial processing occurs on the nearest processing date before the exclusion date.

For example, if cash payments for benefit cases are normally made on a Monday and next Monday is a public holiday then next Monday can be marked as an exclusion date on the cash payment financial calendar. Payment is then made on the nearest valid processing date before the exclusion date.

A separate financial calendar is provided for each delivery method that the organization uses. A separate financial calendar is useful because different delivery methods might have different exclusion dates. For example, if the organization cannot issue checks on public holidays, it might be capable of processing electronic fund transfer (EFT) transactions.

Payment exclusion dates can also be used with cover period offsets. For example, a Tuesday payment date is processed on the Friday in advance due to a two-day in advance cover pattern offset and payment exclusion date settings for the Saturday and Sunday.

**Note:** Exclusion dates do not apply to online payments because online payments are front office payments that are a specific payment for a specific case on a specific date.

### **Related concepts**

[Issuing payments online and to third parties on page 20](#)

Organizations can issue payments online or to third parties, for example utility payments can be issued to utility participants.

## **Financial instruction line items and financial instructions**

A financial instruction line item (ILI) is the most basic financial processing element within the application. At least one instruction line item is generated for every financial process that occurs.

### **Financial instruction line items**

Instruction line items for a case are created when financial components are processed. Where a financial component is a schedule to pay or bill, an instruction line item for a case is an instance of that schedule, for example a payment or bill.

For example, a financial component schedules that John Smith is paid \$25 by check every week, starting on 1 January. On 1 January, the financial component is processed into a financial instruction line item. After that, a financial instruction line item is created every week until the end date of the financial component. Information that is transferred from the financial component includes the nominee, the delivery pattern, the amount, and the period the payment or bill covers.

### **Financial instructions**

A financial instruction is a rolled up view of one or more instruction line items. The rolling up of instruction line items into a single financial instruction consolidates a nominee's instruction line items of the same category into a single financial instruction. Rolling up is also used to consolidate a number of payments or liabilities for a nominee.

For example, an instruction that represents four separate payment instruction line items might be created and issued to a nominee as a single payment. Rolling up is useful because it simplifies the organization's financial interactions with participants. For example, a payment instruction might include the income support instruction line item that is rolled up with any deduction instruction line items, for example third-party deductions.

### Payment groups

You can create a payment group so that the payment instruction line items from certain products or programs are rolled up into one payment instruction. For example, if a nominee is paid \$50, \$75 and \$100 from three different programs and no payment groups are defined, all the payment instruction line items are rolled up into a single payment instruction of \$225. However, if an agency requires that payments from certain programs are rolled up separately, payment groups can be used. If a payment group is created for the programs or products of the first two payments, that is \$50 and \$75, then the system generates two payment instructions, one for \$125 and another for \$100.

You can also create financial instructions and related instruction line items outside of case processing. For example, the following list outlines when financial instructions and related instruction line items are created:

- A payment received is allocated toward a liability.
- A manual payment is captured.
- A liability is written off.

For more information about each financial instruction type and its related financial instruction line items, see the *Financial instruction types* related link.

**Note:** If your organization is using an integrated environment, instruction line items are the financial integration point between the two systems. In this environment, the remaining financial building blocks are not produced by the application. Instead, the building blocks' equivalents are created and managed by the Enterprise Resource Planning (ERP) financial system.

For more information about integration with an ERP financial system, see the *Financial Adapter Technical Overview Guide*.

### Related concepts

[Financial instruction types on page 41](#)

Financial processing uses different financial instruction types to process instruction line items.

### ***Determining the roll-up of financial instruction line items***

Five pieces of information determine how financial instruction line items are rolled up into a financial instruction.

- **Financial instruction line items of the same category**  
Financial instruction line items of the same category can be rolled up into the same financial instruction. For example, when taxes are applied to a benefit, both the benefit instruction line item and the tax instruction line item use the same category of benefit. So, the related instruction line items can be rolled-up and processed together.

- **A financial instruction is issued to a single nominee**  
When a financial instruction is issued to a single nominee, all instruction line items must share the nominee.
- **All instruction line items must have the same delivery method and currency**  
To process a financial instruction, all instruction line items must have the same delivery method and currency.
- **Determine the financial instruction line items are ready to be processed at the same time**  
The financial instruction line item cover period is used to determine the financial instruction line items are ready to be processed at the same time and so, can be rolled up into one financial instruction.
- **A credit or a debit**  
Each financial instruction line item has an amount that is either a credit or a debit. The amount is used to calculate the overall value of a financial instruction. For example, a payment instruction that includes a payment for \$100 and a tax deduction of \$10 results in a payment amount of \$90.

### ***Differentiating between the due date and processed date***

All financial instructions have a due date and a processed date.

#### **Due date**

The due date is the date that the nominee is scheduled to receive a payment or invoice. For example, for an electronic funds transfer, the due date is the date that the payment is scheduled to be deposited in the nominee's bank account and available to be withdrawn. As part of financial component processing, the due date is calculated by using the nominee's delivery pattern. For example, if a client is paid every week on a Monday in advance, the due date is always a Monday, that is the day that the client is always due to be paid or invoiced.

#### **Processed date**

To ensure that the nominee receives the payment in time for the due date, any payments or invoices must be processed before the financial processing date closes. In the preceding example, the financial processing date in the delivery pattern closes on a Thursday. This is the offset period that is required to ensure that the money can get to the nominee's bank account by the following Monday. On Wednesday, the client notifies a change in their circumstances that took effect from the Monday of this week. This change causes a reassessment that results an underpayment to the client. The underpayment covers the period from Monday to Friday of this week because the client is paid weekly in advance. The amount that the client is underpaid is 'due' on the Monday at the start of this week. However, because this date has already past, the client can't receive this payment on this date.

However, when changes occur that impact when payment is physically made, it is important to note that the 'processed date' must also be considered when financial transactions are viewed. The caseworker can use the processed date to reconcile that a due date might not have been met for the payment, where the processed date does not allow enough time for the due date to still be met. In the preceding example, the due date is Monday, 13 November. However, the actual processed date is Wednesday, 15 November, which falls in time for the financial processing of the following Monday's payment, it reveals that payment date will be the Monday, 20 November.

## Financial instruments and payslips

Financial instruments are the records of actual payments, liabilities, or payments received that organizations issue or receive. Payslips are a physical record of the financial instruction line items that pertain to a particular payment or liability instruction that is used for record keeping purposes.

- **Financial instruments**

Financial instruments are recorded for financial instructions. For example, a payment instrument is recorded when a payment instruction is issued for a person.

- **Payslips**

Payslips can be issued to a participant, case nominee, or third party. Each payslip contains a header with recipient details, payment or liability details, and a section that contains instruction line item details for each instruction line item in the payment or liability.

### ***1.2 Generating payments and liabilities***

---

Payments and liabilities due for active cases are automatically generated when the financial batch suite is run. The financial batch suite is a group of four processes that run in sequence. Running these processes at regular intervals automates payment and liability generation and ensures that eligible participants are paid or billed according to their delivery frequency.

The following diagram provides a schematic overview of the processes within the financial batch suite. The following list outlines the order of the processes within the financial batch suite:

1. Determine product delivery eligibility batch process runs to generate financial components.
2. Generate instruction line items batch process runs to generate instruction line items.
3. Generate instruments batch process runs to generate financial instructions and financial instruments.
4. Generate payslips process runs to generate payslips for all financial instruments.

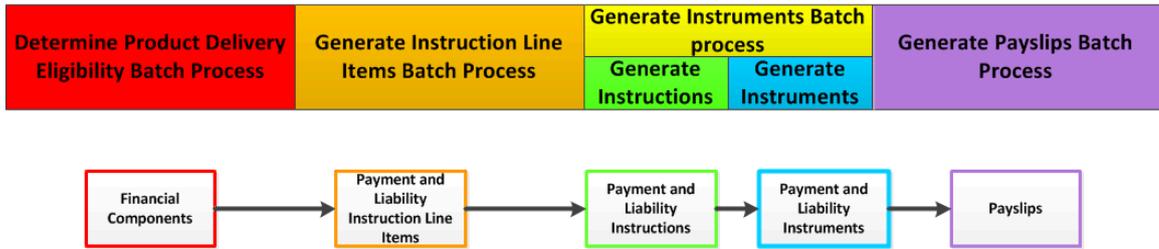


Figure 1: The financial batch suite

**Note:** If your organization is using an integrated environment, the steps that are described in the diagram are not all applicable. When the generate instruction line items batch process is run, the transfer instruction line items batch process, is run. The batch job transfers the instruction line items to the Enterprise Resource Planning (ERP) financial system for further processing.

For more information, see the *Financial Adapter Technical Overview Guide*.

## Simulating payments

The system simulates payments to indicate how much a participant is due to receive.

- **Simulating payments and deductions**

Users can simulate online payments. By simulating a payment, a user can view all payments that are due to a participant for a specific date before the payment is generated and issued. By simulating a payment, the user can view how much money a participant is due to receive over a specific period.

The system also displays any deductions due on payments. If the user is not satisfied with the details of a deduction that is due to be processed, the deduction can be modified to achieve the intended result.

- **Generating a simulation of the payment**

When a payment is simulated, the system retrieves the financial components that are created when the case was activated, or last reassessed, and rolls them forward until the date that is entered by the user is reached. The system then generates a simulation of the payment for that week or for the delivery period of the case. The system also calculates the amount, cover period, and effective date of the payment. If deductions are set up on the case, these deductions are applied to the financial components and the reduced payment amount is also

displayed. If no deductions exist on the case, the system displays only the payments due on that particular date.

A history of simulated payment records is maintained over time to provide agency workers with a view of potential payments with deductions to clients.

## Creating and expiring financial components

Eligibility and entitlement processing creates financial components from eligible case components. A financial component is expired when it reaches its end date.

- **Creating financial components**

When eligibility determination processing is initiated for a case, either through batch processing or online, eligibility is determined for each case component, for example an income support benefit.

Financial components are created for each case component for which the client is eligible using the delivery pattern of the nominee assigned to each case component. Secondary financial components for every deduction that is set up on the case are also created.

For more information about the eligibility and entitlement engine, see the *Inside Eligibility and Entitlement Using Cúram Express Rules Guide*.

- **Expiring financial components**

One-off financial components expire immediately after the financial component is processed. For recurring financial components, the financial component remains live, and continues to be processed, until the end date is reached. The component is then expired.

A financial component is also expired if reassessment finds the primary client ineligible for a case component that is associated to the financial component. This can occur anytime the financial component is reassessed and helps to ensure that a financial component is not processed if a change in circumstances affects the primary client's eligibility.

## Generating financial instruction line items

Financial components are processed into financial instruction line items when the generate instruction line item batch process is run.

The financial components for a single benefit case can also be processed into instruction line items as part of generating payments online. Reassessment is always performed before a financial component is processed. Reassessment ensures that the primary client is still eligible and that the financial component is still valid. A single financial instruction line item is generated when a one-off financial component is processed. In contrast, a financial instruction line item is generated every time a recurring financial component is processed.

Before a financial component is processed into an instruction line item, the system checks for any payment exclusion dates that are set up for the delivery method. If any payment exclusion dates are set up for the delivery method, the processed date for the instruction line item is adjusted. For example, if the delivery method is cash and the financial component due date is on a payment exclusion date for the cash delivery method, the system adjusts the financial instruction line item processed date to a valid processing date.

**Note:** If your organization is using an integrated environment, after the instruction line items are generated and transferred, the remaining processing is carried out by the Enterprise Resource Planning (ERP) financial system.

For more information, see the *Financial Adapter Technical Overview Guide*.

## Generating financial instructions

Financial instructions for benefit payments and liabilities are created when the batch process runs. The financial instructions for a single benefit case can also be created as part of generating payments online.

- **Creating financial components**

Payment and liability financial instructions are created when one or more instruction line items are rolled up into a single financial instruction. Only instruction line items with the same category, nominee, delivery pattern, and currency can be rolled up together. For example, two payment instruction line items that are generated from the same financial component over two processing dates might be rolled up into a payment instruction. Or a payment instruction line item might be rolled up with a deduction or tax adjustment line item. For more information, see the *Processing deductions and adjustments* related link.

If the agency does not want to roll up all the payment instruction line items from different programs into a single instruction, the agency can add programs or products to a payment group. Adding programs or products to a payment group ensures that only payment instruction line items that belong to a payment group are rolled up into a single instruction.

- **Example**

A nominee is paid \$50, \$75, and \$100 from three different programs. If a payment group is not defined, all the payment instruction line items are rolled up into a single payment instruction of \$225. However, if it is required that the payment of \$50 from a particular program is not to be rolled up with payments from the other programs, the following list outlines the actions that the agency must take:

1. Create a payment group.
2. Add the program or product whose payment is not to be rolled up with payments from other programs. As a result, the system generates two payment instructions: one payment instruction for \$50 and another payment instruction for \$175.

- **Rolling up the instruction line item into the instruction**

A payment or liability instruction is created when the first instruction line item to be rolled up into the instruction is processed. Depending on the type of the remaining instruction line items, the amount of the instruction is then added to or subtracted from. For example, if a payment instruction line item is rolled up into a payment instruction, the amount of the payment instruction is lessened when a related deduction instruction line item is rolled up into the same payment instruction.

When a payment instruction line item is rolled up, its outstanding amount is updated to zero and its status is changed to processed. The changes represent the fact that the instruction line item is processed into a payment instruction, and that a payment is issued to the case nominee.

When a liability instruction line item is rolled up, its status is also changed to processed. However, the outstanding amount of the liability's instruction line item remains equal to the full amount of the liability to indicate that the liability is still outstanding and that the amount is still owed to the organization. The outstanding amount is only reduced or cleared when the liability is written off, reversed, or allocated against. For more information, see the *Maintaining payments and liabilities* related link.

### **Related concepts**

#### [Processing deductions and adjustments on page 23](#)

Deductions, tax, and surcharge adjustments are processed when payments and liabilities are generated. Deductions and tax adjustments are applied to benefits while surcharge adjustments are applied to liabilities. Financial account adjustments can be made to a participant's financials account.

#### [Maintaining payments and liabilities on page 32](#)

Payment and liability processing requires continuous maintenance to capture how client circumstances change over time. Maintenance functions for payments include the ability to capture manual payments, invalidate payments, cancel and reissue payments, and approve suspended payments. Maintenance functions for liabilities include the ability to reverse and to write off liability instructions.

## **Generating financial instruments and payslips**

Financial instruments are generated when the generate instruments batch process is run. To generate payslips, run the generate payslips batch process.

- **Generating financial instruments**

The generate instruments batch process is run directly after the generate instruction line items batch process.

To allow for integration with external financial systems, payment and billing information from financial instrument records is extracted to an external file. For example, payment information for instruments with a delivery method of "check" might be extracted to a system to draft checks.

- **Generating payslips**

If payslips are required, run the generate payslips batch process after the generation of instruments. The type of payslip that is generated depends on the type of recipient, for example the primary client or a utility.

## **Issuing payments online and to third parties**

Organizations can issue payments online or to third parties, for example utility payments can be issued to utility participants.

- **Issuing payments online**

Payments can be issued online for a single benefit case without calling the financial batch suite. Issuing payments like this might be necessary if the primary client needs emergency assistance. Payments can be issued online for active cases only.

**Note:** If your organization uses an integrated environment, issuing payments online is coordinated across both systems.

For more information, see the *Financial Adapter Technical Overview Guide*.

- **Issuing payments to third parties**

Third-party payments are processed according to the third-party participant's preferred payment method, payment frequency, and next payment date. The payment method defines the method of payment, for example, cash, check, and so on. The payment frequency defines the frequency with which payments are issued. The next payment date is calculated by adding the payment frequency to the previous payment date. A preferred currency can also be set for each participant. The setting defines the currency in which third-party payments are issued to the participant.

Third-party payments are issued when the issue concern payments batch process is run. The batch process searches for all participants due for payment. It processes payments for each of the participants and issues payments according to the payment method defined for each. As part of issuing concern payments, the system checks for any payment exclusion dates that are set up for the third-party participant's delivery method and adjusts the payment date as required. For more information about payment exclusion dates, see the *Payment exclusion dates for delivery methods* related link.

The next payment date is then rolled forward to the next valid payment date. For example, a utility participant might specify that it wants to receive payments from the organization every quarter by electronic fund transfer (EFT). Such information is stored on the utility participant record. Every quarter, a payment is issued by EFT. The next payment date is then rolled forward to the appropriate date in the next quarter.

### Related concepts

[Payment exclusion dates for delivery methods on page 13](#)

Payment exclusion dates represent the days on which the organization cannot make payments that use a particular delivery method. A prepayment requirement specifies that financial processing occurs on the nearest processing date before the exclusion date.

## Examples of generating payments

As an example of generating payments, John Smith, who is the primary client and case nominee, is eligible to receive a \$35 payment each week for one month. The payments are issued for the case component, maximum personal benefit.

The following sections list the processing that occurs at each stage of financial processing for the example.

- **Creating financial components**

When the determine product delivery eligibility batch process is run, John Smith is found eligible for the maximum personal benefit from 1 February 2005 to 28 February 2005. The system creates a recurring financial component. The following list outlines the details of the recurring financial component:

- Case nominee: John Smith.
- Amount: \$35.

- Delivery method: Check.
- Delivery frequency: Weekly in advance.
- Category: Payment.
- Type: Maximum personal benefit.
- Start Date: 1 February.
- End Date: 28 February.
- Expiry Date: 21 February.
- First Due Date: 1 February.

- **Generating financial instruction line items**

On 1 February, the generate instruction line items batch process runs and the system searches for any financial components with a processing date on or before 1 February. John Smith's financial component reaches its first due date, so the system creates an instruction line item. The following list outlines the information that is stored on the instruction line item:

- Case nominee: John Smith.
- Delivery method: Check.
- Category: Payment.
- Type: Maximum personal benefit.
- Cover period = 1 February to 7 February.
- Amount: \$35.
- Unprocessed amount: \$35
- Status: Unprocessed.
- Processed date: 1 February.

- **Generating a financial instruction**

When the generate instruments batch process is run, the instruction line item for 1 February to 7 February is rolled up into one payment financial instruction. The following list describes the payment financial instruction:

- Case nominee: John Smith.
- Delivery method: Check.
- Category: Payment instruction.
- Total amount: \$35.
- Cover period: 1 February to 7 February.

The status of the payment instruction line item is updated to processed and the outstanding amount is updated to zero.

- **Generating a payment instrument**

When the generate instruments batch process is run, the system creates a payment instrument from the payment financial instruction. The following list describes the payment instrument:

- Case nominee: John Smith.
- Delivery method: Check.
- Category: Payment instruction.
- Total amount: \$35.
- Cover period: 1 February to 7 February.

- **Generating a payslip**

When the generate payslips batch process is run, the system can generate a payslip for the payment financial instruction. The following list describes the payslip:

- Nominee name: John Smith.
- Component type = Maximum personal benefit.
- Cover period from: 1 February.
- Cover period to: 7 February.
- Amount: \$35.
- **The next processing dates**  
On 8 February, the system creates another instruction line item with a cover period 8 February to 14 February. The next processing date is then rolled forward to 15 February. The system creates a financial instruction, instrument, and payslip for the new instruction line item. The processing repeats on every processing date until the recurring financial component is expired.

## 1.3 Processing deductions and adjustments

---

Deductions, tax, and surcharge adjustments are processed when payments and liabilities are generated. Deductions and tax adjustments are applied to benefits while surcharge adjustments are applied to liabilities. Financial account adjustments can be made to a participant's financials account.

### Deductions

Deductions allow the organization to allot part of a person's benefit payments to a specific purpose. This purpose is dependent on the category of deduction. There are three deduction categories within the application: applied deductions, un-applied deductions, and third party deductions.

For applied deductions, the amount is deducted from the benefit and applied toward any participant's outstanding liability. For example, \$10 of a person's weekly benefits can be applied toward paying off an overpayment that was previously issued to him or her. A payment can be reissued with an applied deduction if it is cancelled.

For un-applied deductions, the amount is also deducted from the benefit. It is then saved on the system as an unprocessed instruction line item payable to the organization. Un-applied deductions are used to recoup monies previously paid out by the organization. For example, if a person was previously issued money out of an emergency fund, deductions could be made from the person's benefit payments as a means of recouping the money for the organization. A payment can be reissued with an un-applied deduction if it is cancelled.

Third party deductions allow a portion of a person's benefit to be directed toward bills, charges, or debts owed to a third party. Third party payments are made to any other participants registered on the system. For example, a person can use a portion of a benefit to pay his or her electricity bills. The electricity provider is a registered utility on the system.

Deduction are set up as part of case administration. A complete history of deductions, active and inactive, is maintained at the case level.

For more information about setting up and maintaining deductions, see the *Deductions Guide*.

### **Creating secondary financial components for deductions**

Once the primary financial components have been created, secondary financial components are created for each deduction existing on the case. If a deduction has been created against a particular benefit component, the deduction financial component is associated with the benefit financial component.

### **Generating instruction line items for deductions**

Financial instruction line items are generated for all three deduction categories (applied, un-applied, third party) by the processing of secondary financial components. Secondary financial components are processed as part of the generate instruction line items batch process.

This occurs after all payment instruction line items have been generated, i.e., after the primary financial component has been processed. The amount of the deduction is checked against settings that are configured as part of system administration. Based on this comparison, the system determines whether or not the secondary financial components are processed.

For example, before processing the secondary financial components, the system determines whether or not there are sufficient funds in the payment amount to cover the deduction amount. All deductions have a setting that determines the action to take if there are insufficient funds available to process the full deduction amount. Depending on this setting, part of a deduction may be processed against the amount that is available or the deduction may not be processed at all.

Deductions can be prioritized by the organization in order of importance. For example, it may be more important to process a deduction that is used to pay housing costs than a deduction that is used to repay a liability owed to the organization. Based on its priority, each deduction is applied to the total deductible amount calculated for the nominee who receives the benefit payment. For each applied, un-applied, or third-party payment deduction that is processed, two line item records are created. The first of these is a debit against the benefit. As described in the following section, the second is applied toward a liability, stored on the system, or paid toward a third party such as a utility, depending on its type.

Configuration is also available which provides the ability for the agency to define whether or not overlapping deductions are allowed. If a deduction is configured to prevent overlapping deductions, a validation will be displayed if a user tries to activate a deduction which already exists on the case for an overlapping time period. This can be configured for all categories of deductions (applied, un-applied and third party).

The two instruction line items created are related to each other by a relationship record that is automatically created by the system. This allows for traceability back to the case nominee whose payment the deduction was made from. For example, when a bulk payment is issued to a utility (see below), each line item in that payment will be traceable back to a case nominee.

### **Processing deductions**

The first instruction line item created for all deduction categories is rolled up into the payment instruction for the benefit. This means that the person in receipt of benefits will receive a payment less the deduction amount. For example, a utility deduction amount of \$5 might be subtracted from a payment of \$35 so that the payment instruction for the benefit would be for the total amount of \$30.

For applied deductions, the second instruction line item will be rolled up into a payment received instruction that is allocated toward the outstanding liability. Note that the allocated payment received instruction is system-generated, but that it functions identically to an allocated payment received from outside the system. For more information about allocating payments received, see the *Allocating payments received* related link.

For un-applied deductions, the second instruction line item is saved on the system as a line item payable to the organization.

For third party deductions, the third party payment instruction line item is issued to the relevant participant company by calling the Issue Concern Payments batch process. This batch process is used to issue payments to participants. When run for a participant, it searches for all unprocessed third party payment instruction line items for the participant and rolls them up into one payment instruction. This allows the organization to issue a single payment to a participant in respect to deductions made from multiple cases over a period of time. For example, the organization may issue a utility a payment once per quarter.

For more information about deduction processing, see the *Deductions Guide*.

### **Related concepts**

[Allocating payments received on page 38](#)

A payment received can be allocated toward any number of outstanding liability instructions by the person or employer from whom the payment was received.

## **Tax adjustments**

Benefits can be adjusted to take relevant taxes into account.

For example, a 10% tax may be deducted from every payment. A setting at the product level indicates whether adjustments are required for a product. If the setting is turned on for a benefit product, taxes will be applied to all payments issued in respect of the product.

The adjustment rate for taxes can be maintained as part of rate table administration. The same rate will be applied to all payments. For example, a tax of 5 percent may be applied to all payments for a benefit product. When financial components are created for a benefit, information regarding whether or not tax adjustment processing is required is saved on each of the financial components.

### ***Generating instruction line items for a tax adjustment***

When the generate instruction line items batch process is run, or when payments are generated online, the system determines whether or not taxes should be applied to the benefit.

If taxes should be applied, the system creates two instruction line items. The first of these is a debit against the benefit. As described in the following section, the second is used to pay the tax authority.

The two instruction line items created for a tax adjustment are related to each other by a relationship record. This allows for traceability back to the case nominee from whose payment the tax adjustment was made. For example, when a bulk payment is issued to a tax authority (see below), each line item in that payment will be traceable back to a case nominee.

### ***Applying taxes***

The first instruction line item created for the tax adjustment is rolled up into the payment instruction for the benefit.

This means that the person in receipt of benefits will receive benefits less the amount of the tax adjustment. For example, a benefit with a payment amount of \$50 might be adjusted by a 10% tax rate so that the payment instruction for the benefit would be for a total of \$45.

The second line item for the tax adjustment is issued to the tax authority by calling the Issue Concern Payments batch process (note that tax authorities are registered as service supplier participants). The system searches for all unprocessed tax payment instruction line items and rolls them up into one payment instruction for the tax authority. This allows the organization to issue a single payment to a tax authority for deductions made from multiple cases over a period of time. For example, the organization may issue a tax authority a payment once per tax year.

## **Surcharge adjustments**

Surcharges are additional charges applied to liabilities that have not been processed within an appropriate time period. For example, if an employer is issued a bill and does not pay all or a portion of this bill within the appropriate time period, a surcharge is applied to the outstanding amount. The employer is issued a new bill for the surcharge.

- **Generating instructions for surcharge adjustment**

Every time the generate instruments batch process is run, the system searches for all existing instruction line items with outstanding amounts greater than zero. It then determines whether or not surcharge adjustment processing is required for each instruction line item and whether the instruction line item is due for surcharge processing. If these three factors are met, the system applies the surcharge rate to the unprocessed liability amount in order to determine the surcharge amount. An instruction line item is then created for the surcharge adjustment.

When the generate instruments batch process is run, the system searches for all surcharge instruction line items for the same client and rolls these instruction line items up with any additional liability instruction line items that the client holds. For example, if a surcharge instruction line item is created for an employer, this will be rolled up with any other liability line items existing for the employer when the generate instruments batch process is run.

A setting at the product level indicates whether adjustments are required for a product. If the setting is turned on for a liability product, surcharges will be applied to bills if they remain outstanding for one month. If a bill is not cleared within one month, it is surcharged at the adjustment frequency configured at the product level. The adjustment rate for surcharges is set at a fixed rate. For example, the organization may specify that liabilities left unpaid for one month will be surcharged at five percent.

## **Financial account adjustments**

Adjustments can be used to correct a client's financial account so that the system does not attempt to generate a compensating over or under payment. For example, the organization might need to write a check manually to the client and can balance the client's account through an adjustment.

A client's financial account can be adjusted by applying a debit or credit amount to the account.

Adjustments do not result in any payment or liability that is issued to the client and are purely to balance an account.

## Examples of payment generation with deductions and tax adjustments

Use the eight examples of payment generation with deductions and tax adjustments for reference.

### Example

The primary client (and case nominee), John Smith, is eligible to receive a \$35 check one time a week. This is for the case component, Max Personal Benefit. John Smith is eligible for this payment from 1 February to 7 February, that is, one week. A third-party deduction is also set up on the Max Personal Benefit case component with an amount of \$5. This deduction is to be paid toward John Smith's electricity bills. Additionally, tax adjustment processing is set up on the benefit.

- **Creating financial components**

John Smith is found eligible for the Max Personal Benefit from 1 February 2005 to 7 February 2005. The following list outlines the details of the created financial component:

- Nominee: John Smith.
- Amount: \$35.
- Delivery method: Check.
- Delivery frequency: Weekly in advance.
- Category: Payment.
- Type: Max personal benefit.
- Start date: 1 February.
- End date: 7 February.
- Expiry date: 1 February.
- Processing date: 1 February.
- Tax adjustment: True.

The following list outlines the details of the secondary financial component that is created for the third-party deduction:

- Nominee: John Smith.
- Amount: \$5.
- Delivery method: Check.
- Delivery frequency: Weekly in advance.
- Type: Deduction payment.
- Start date: 1 February.
- End date: 7 February.
- Expiry date: 1 February.

- **Generating payment financial instruction line item**

On 1 February, the primary financial component reaches its processing date, and an instruction line item is created. The following list outlines the information that is stored on the instruction line item:

- Nominee: John Smith.
- Amount: \$35
- Delivery method: Check.
- Category: Payment
- Type: Max personal benefit.
- Cover period: 1 February to 7 February.
- Unprocessed amount: \$35
- Status: Unprocessed.
- Tax adjustment: True.

After the instruction line item is created, the financial component is expired, as there is no next processing date.

- **Generating third-party financial instruction line items**

When the primary financial component is processed into the payment instruction line item, the secondary financial component is also processed and a third-party payment and a third-party deduction instruction line item are created. The following list outlines the third-party deduction instruction line item:

- Nominee: John Smith.
- Amount: \$5.
- Delivery method: Check.
- Category: Payment.
- Type: Deduction item.
- Cover period: 1 February to 7 February.
- Unprocessed amount: \$5.
- Status: Unprocessed.

The following list outlines the third-party payment instruction line item:

- Nominee: The Electric Company.
- Amount: \$5.
- Category: Third-party payment.
- Type: Deduction payment.
- Cover period: 1 February to 7 February.
- Unprocessed amount: \$5.
- Status: Unprocessed.

- **Generating tax financial instruction line items**

The tax rate is set to 10%. A tax deduction and a tax payment instruction line item are created. The following list outlines the tax deduction instruction line item:

- Nominee: John Smith.
- Amount: \$3.50.
- Delivery method: Check.
- Category: Payment.
- Type: Tax deduction.
- Cover period: 1 February to 7 February.

- Unprocessed amount: \$3.50.
- Status: Unprocessed.

The following list outlines the tax payment instruction line item:

- Nominee: Tax authority.
  - Amount: \$3.50.
  - Delivery method: Check.
  - Category: Tax payment.
  - Type: Tax payment.
  - Cover period: 1 February to 7 February.
  - Unprocessed amount: \$3.50.
  - Status: Unprocessed.
- **Generating a payment instruction**

The payment instruction line item, third-party deduction instruction line item, and tax deduction instruction line item for 1 February to 7 February are rolled up into one payment instruction. The following list outlines the payment financial instruction:

- Nominee: John Smith.
- Delivery method: Check.
- Category: Payment
- Total amount =  $\$35 - \$5 - \$3.50 = \$26.50$ .
- Cover period: 1 February to 7 February.

Here the status of the three instruction line items is updated to processed and their outstanding amounts are updated to zero.

**Note:** If your organization is using an integrated environment, payment instructions are not generated as described. The equivalent processing is carried out by the Enterprise Resource Planning (ERP) financial system.

For more information, see the *Financial Adapter Technical Overview Guide*.

- **Generating payment instructions for the third party and tax authority**

The tax payment instruction line item and the third-party payment instruction line item are rolled up into payment instructions for the tax authority and the electricity company, respectively. This is done when the Issue Concern Payments batch process is run. Note that this batch process is usually run across many cases so that the third party and tax authority would receive single payments for all deductions that are made over a period. Note that the issue concern payment batch process also generates payment instruments and payslips for the third party and tax authority, respectively.

**Note:** If your organization is using an integrated environment, payment instructions for a third party and the tax authority are not generated as described. The equivalent processing is carried out by the Enterprise Resource Planning (ERP) financial system.

For more information, see the *Financial Adapter Technical Overview Guide*.

- **Generating a payment instrument**

A payment instrument is created from the payment instruction for the benefit when the Generate Instruments batch process is run. This instrument includes the information that is described for the payment financial instruction.

**Note:** If your organization is using an integrated environment, payment instruments are not generated as described. The equivalent processing is carried out by the Enterprise Resource Planning (ERP) financial system.

For more information, see the *Financial Adapter Technical Overview Guide*.

- **Generating payslips**

A payslip can be generated for the payment instrument for the benefit. The following list outlines the information included in the payslip:

- Nominee: John Smith.
- Component type: Max personal benefit.
- Cover period from: 1 February.
- Cover period to: 7 February.
- Amount: \$26.50.

**Note:** If your organization is using an integrated environment, payslips are not generated as described. The equivalent processing is carried out by the Enterprise Resource Planning (ERP) financial system.

For more information, see the *Financial Adapter Technical Overview Guide*.

## Examples of liability with surcharge adjustments

Use the five examples of liability with surcharge adjustments for reference.

### Example

The primary client, and case nominee, is the employer, Midway Emporium. Midway Emporium is liable for service fees of the amount of \$100 per month from January 2005 through February 2005.

Additionally, surcharge adjustment processing is set up on the liability. The surcharge rate is 10%. The time limit for paying the liability is one month. Surcharges are applied to liability instruction line item that go unprocessed for more than one month.

- **Creating a financial component**

When the Determine Product Delivery Eligibility batch process is run, Midway Emporium is determined to be liable for service fees in the amount of \$100 per month from January 2005 through February 2005. The following list outlines the details of the recurring financial component that is created:

- Nominee: Midway Emporium.
- Amount: \$100.
- Delivery frequency: Monthly in advance.

- Category: Liability.
  - Type: Service fee.
  - Start date: 1 January.
  - End date: 28 February.
  - Processing date: 1 January.
  - Surcharge adjustment: True.
- **Generating a liability financial instruction line item**  
When the generate instruction line items batch process is run, the system searches for any financial components with a processing date on or before the current date. On 1 January, the financial component reaches its first processing date, and an instruction line item is created. The following list outlines the details of the instruction line item:

- Nominee: Midway Emporium.
- Amount: \$100.
- Delivery method: Invoice.
- Category: Liability.
- Type: Service fees.
- Cover period: 1 January - 31 January.
- Unprocessed amount: \$100.
- Status: Unprocessed.
- Surcharge adjustment: True.

After the instruction line item is created, the next processing date on the financial component is rolled forward to 1 February.

- **Generating a liability instrument**  
When the generate instruments batch process is run, the instruction line item for 1 January - 31 January is rolled up into one liability financial instruction. The following list outlines the liability financial instruction:

- Nominee: Midway Emporium.
- Delivery method: Invoice.
- Category: Liability.
- Total Amount: \$100.
- Cover period: 1 January - 31 January.

Here, the status of the liability instruction line item is updated to processed. The outstanding amount remains at \$100 until allocations are made toward it.

**Note:** If your organization is using an integrated environment, liability instructions are not generated as described. The equivalent processing is carried out by the Enterprise Resource Planning (ERP) financial system.

For more information, see the *Financial Adapter Technical Overview Guide*.

- **Generating a liability instrument**  
When the generate instruments batch process is run, a liability instrument is created from the liability instruction. The following list outlines the liability instrument:

- Nominee: Midway Emporium.
- Delivery method: Invoice.
- Category: Liability.
- Total Amount: \$100.
- Cover period: 1 January - 31 January.

Here, the status of the liability instruction line item is updated to processed. The outstanding amount remains at \$100 until allocations are made toward it.

**Note:** If your organization is using an integrated environment, liability instructions are not generated as described. The equivalent processing is carried out by the Enterprise Resource Planning (ERP) financial system.

For more information, see the *Financial Adapter Technical Overview Guide*.

- **The next processing date**

On 1 February, another liability instruction line item is created with a cover period of 1 February - 28 February. The financial component is expired as this is the last processing date. The system reads that no payments were made toward the liability instruction line item for the month of January. Since it is one month since the instruction line item was issued, surcharges must be applied to the unprocessed liability instruction line item. The surcharge rate is set to 10%. Therefore, a surcharge instruction line item is created. The following list outlines the surcharge instruction line item:

- Nominee: Midway Emporium.
- Amount: \$10.
- Delivery method: Invoice.
- Category: Liability.
- Type: Surcharge.
- Cover period: 1 January - 31 January.
- Unprocessed amount: \$10.
- Status: Unprocessed.

The surcharge instruction line item and the liability instruction line item for February is rolled up into a single liability instruction. The system creates a liability instrument.

## 1.4 Maintaining payments and liabilities

Payment and liability processing requires continuous maintenance to capture how client circumstances change over time. Maintenance functions for payments include the ability to capture manual payments, invalidate payments, cancel and reissue payments, and approve suspended payments. Maintenance functions for liabilities include the ability to reverse and to write off liability instructions.

## Capturing manual payments

A manual payment is a payment to a case nominee that is issued outside of the system, such as when the system is off-line or when a manual payment is issued by an agency not normally connected to the system. If for any reason the system is unavailable, and a payment is issued to a person, then information regarding that manual payment can be captured on the system.

Capturing the manual payment calls processes that recreate all the elements of a payment including the payment instrument, the payment instruction, the rolled up instruction line items, and any relevant deduction/tax instruction line items.

The cover period of a manual payment can be recorded to facilitate the recording of a manual payment for a specific past or future cover period. If the cover period is specified the system will use that cover period in calculating the entitlement amount for a client.

After creating the financial elements of the manual payment, the system then processes the payment due so that the payment is not paid out a second time. The system then compares the amount of the manual payment to the amount of the processed payment and creates an over or under payment to rectify the difference. A notification task is also sent to the case owner notifying him or her that an over or under payment has occurred.

**Note:** If your organization is using an integrated environment, the capture manual payment process is not as described. The equivalent processing is carried out by the Enterprise Resource Planning System (ERP) financial system.

For more information, see the *Financial Adapter Technical Overview Guide*.

## Canceling and reissuing payments

A payment issued in error can be canceled. Canceling a payment indicates that a payment has not been received.

For example, a payment might be canceled if a participant reports that a check has been lost in the mail. A payment is canceled at the payment instruction level. When a payment instruction is canceled, the status of the payment instruction changes from issued to canceled. . All of the instruction line items within the payment instruction are also canceled and their status changes from processed to canceled.

A payment may also be canceled because the bank account has been closed or an error in instrument details may require the cancellation of the payment. When a payment is canceled, the reason for canceling the payment is recorded. On cancellation, the payment instruction is negated rather than deleted from the system. This is for accountability and traceability purposes. A new reversal instruction is created to cancel out the amount of the payment instruction. A reversal instruction line item is also created for every instruction line item that was rolled up into the payment instruction.

Any payment that has been canceled can be reissued to the original nominee or to an alternative nominee. For example, a check payment that has been lost can be reissued to the original nominee. Payments can be reissued using any of the nominee's active delivery patterns. For example, an original payment issued monthly by check can be reissued daily by cash.

A payment that has been canceled can be reissued and have an applied, or unapplied, deduction created against it. This creates a deduction, which is visible to a caseworker as per any other deduction on the system, including a full deduction history. A deduction created when reissuing a payment is processed into the reissued payment, and in the case of an applied deduction, applied against the overpayment. This achieves a reduced payment that is issued to the client for the cover period in question. Invalidating payment with a replacement payment, as described below, is an alternative mechanism to achieve a reduced payment to a client. Whether to use reissue payment with deduction applied or invalidate payment with replacement payment, depends on whether the change in circumstance is recorded before the payment is canceled. For example, if a payment is issued to a client and the client dies during this period. If the agency is first notified of the death before payment being returned (or perhaps payment issuance later fails if, for example, bank account were closed), the date of death is recorded which generates a change in circumstance and hence an overpayment is created. At this point, to get a new payment issued for the period up to the date of death, Invalidate Payment cannot be used as the period has already been reassessed and therefore there is no longer a payment only for the cover period in question; the payment and the reassessment result (payment correction or over/under payment) need to be considered together for this period. In this scenario a reduced payment can be issued for the cover period by canceling the payment and reissuing with a deduction applied to the overpayment case.

When a payment is reissued, new payment instruction and instruction line items are created. The canceled payment instruction and its instruction line items have a status of canceled. The new payment instruction has a status of issued and the new payment instruction line items have a status of processed. A payment can be reissued during the cancellation process or after the payment instruction has been canceled.

**Note:** If your organization is using an integrated environment, the payment cancellation and reissuing process is not as described.

For more information, see the *Financial Adapter Technical Overview Guide*.

## Invalidating payments

Payments issued to clients can be invalidated. The difference between invalidating and canceling a payment is that a canceled payment can be reissued

When a payment is invalidated, it will no longer be considered by system processing. This is to allow for the reissuing of a replacement payment if, for example, the original payment was issued for the incorrect amount. The payment can be both canceled and invalidated at the same time. The system allows for this to happen at time of cancellation, or in two stages: cancel now and invalidate later.

For example, John Smith is receiving benefit payments of \$70 per week, paid in advance on a Monday. He goes back to work on a Tuesday, which means he is only entitled to benefit on the Monday, that is, he is only due \$10 that week. He receives a check for \$70 but phones the organization and informs it of his change of circumstance. The payment is canceled, as he will not be cashing his check of \$70, and invalidated, as the organization wants to issue a replacement payment. The certification on his benefit case is modified to end John Smith's payments on the Monday. This results in a replacement payment for \$10 being issued to him.

Furthermore, if there is a question mark over which day John returned to work when he informs the organization of his change of circumstance, the organization may require documentation detailing the date he started in his new employment. In this instance, the organization can cancel John's payment but not invalidate it until it receives the documentation from his new employer.

When a payment is invalidated, all of its associated components are invalidated with it.

## Approving suspended payments

As part of application administration, a maximum amount can be set up for a delivery method, so that no payments over this amount can be made by the delivery method. Any payments that exceed this maximum amount will be suspended.

For example, the organization may have a policy that no payments over \$100 can be made by cash. If John Smith was scheduled to receive a payment for \$120 in cash, the payment would be automatically suspended until the organization had an opportunity to investigate the reason that the limit was exceeded.

The case nominee will not receive a suspended payment unless the payment is approved by the organization. Approving a suspended payment overrides the maximum amount limitation and allows the payment to be issued. Over and underpayments are still created on cases with suspended payments regardless of whether the suspended payment has been approved or not. This is due to the fact that the over or underpayment may be related to several payments that were made over a period of time, of which some may not be suspended and are therefore valid. Also, the expectation is that agencies will deal with the situation in which a payment has been suspended in a timely manner; otherwise the client will not get paid. If the agency is not in a position to act on a suspended payment quickly or they do not want to create over or under payments on cases with suspended payments, they could choose to suspend the case itself.

**Note:** If your organization is using an integrated environment, the approving suspended payments process is not as described.

For more information, see the *Financial Adapter Technical Overview Guide*.

## Reversing liabilities

All or portions of liabilities can be reversed.

For example, a liability instruction issued to a nominee in error can be reversed such that the nominee is no longer responsible for the full amount owed on the liability. Alternatively a portion of a liability can be reversed, e.g., one liability instruction line item as part of a liability instruction can be reversed, such that the nominee is no longer responsible for the reversed line item but is still responsible for the outstanding amount on the liability instruction.

When a liability instruction line item is reversed, the liability amount is negated rather than deleted from the system. This is for accountability and traceability purposes. A new reversal instruction line item is created to cancel the amount of the liability instruction line item. The reversal instruction line item is automatically allocated toward the liability instruction line item.

This updates the outstanding amount of the liability instruction line item to zero. The reversal instruction line item is rolled up into a reversal instruction.

If the liability instruction line item was allocated toward before it was reversed, these allocations are once again made available to be applied toward other outstanding liabilities. In order to make this possible, the system automatically creates a reversal instruction line item for each allocation and rolls up the instruction line item or items into a reversal instruction. These reversal instruction line items can be allocated toward any of the nominee's remaining liabilities.

**Note:** If your organization is using an integrated environment, the liability reversal process is not as described.

For more information, see the *Financial Adapter Technical Overview Guide*.

## Writing-off liabilities

All or a portion of a liability instruction can be written off so that the nominee is no longer responsible for the amount that has been written off.

For example, if a person declares bankruptcy, all or a portion of that person's debt to the organization can be written off.

The amount of a write-off can be equal to or less than the outstanding amount of the liability instruction. The value of the outstanding amount on the liability will be displayed in the write-off instruction.

When a liability instruction is written off, a write-off instruction line item is created to reflect the write-off amount. This creates a record of the write-off that is useful for accountability and traceability purposes. The write-off instruction line item is automatically allocated toward the liability instruction. This reduces the outstanding amount of the liability instruction by the amount of the write-off.

Write-off instruction line items are rolled up into write-off instructions. This is true for all write-offs regardless of whether an outstanding amount remains on the liability instruction.

**Note:** If your organization is using an integrated environment, the liability write-off process is not as described.

For more information, see the *Financial Adapter Technical Overview Guide*.

## Over and under payment processing

Case reassessment checks if changes in case circumstances may have resulted in a nominee being over or under paid.

An over or under payment occurs when a new decision is created for a period that overlaps with an existing decision or decisions that have already been processed for payment or billing and that new decision differs from the existing decision or decisions. To determine the value of an over or under payment, financial data is created for the new decisions that represents what the

financial components would have been if based on the changes in circumstance. This financial data is compared to the actual processed instruction line items.

For more information, see the *Financial Adapter Technical Overview Guide*.

For more information about case reassessment, see the *Integrated Case Management Guide*.

For example, John Smith received a payment in the amount of \$70. This payment instruction included five benefit instruction line items, three for the child benefit component, and two for the max personal component. New evidence was recorded after this payment instruction which resulted in new decisions. The decision information overlapped with the processed financial components, and showed different amounts that should have been paid for the child benefit and max personal components based on the changes in circumstances. The below table demonstrates the granular representation of over and under payment processing, where each 'actual amount' is a processed instruction line item and each 'reassessed amount' is financial data representing what should have been paid or billed based on the changes in circumstance.

The table describes the granular representation of over and under payments.

Table 2: Granular representation of over and under payments.

From Date	To Date	Component	Actual Amount	Reassessed Amount	Difference
16/02/2009	22/02/2009	Child Benefit	40.00	10.00	-30.00
23/02/2009	01/03/2009	Max Personal	5.00	20.00	15.00
23/02/2009	01/03/2009	Child Benefit	40.00	10.00	-30.00
02/03/2009	08/03/2009	Max Personal	5.00	20.00	15.00
02/03/2009	08/03/2009	Child Benefit	40.00	10.00	-30.00
				<b>Total Over or Under Payment</b>	<b>-60.00</b>

### What can I configure or customize?

You can configure the system to generate an overpayment or an underpayment in a net-zero scenario. To configure the system in this way, log on as a system administrative user and perform the following steps:

1. Click **Application > Miscellaneous**.
2. Set the `curam.miscapp.displayzerodifference` application property to **YES**. The default value is **NO**.
3. To publish the property change, click **Publish**.

## 1.5 Processing and maintaining payments received

A payment received is an amount of money received by the organization and recorded on the system. Payments received are usually sent to the organization in response to a bill. That is,

they are used to pay off liabilities. Unlike payments and liabilities, payments received are not generated as part of case processing. Instead, they are recorded on the system either manually by a user or by using a batch process that records payments received in bulk, for example, all payments received by electronic funds transfer (EFT) from a particular bank. Payments received are associated with a person's financials through the allocation of the payment received toward one or more outstanding liabilities.

**Note:** If your organization is using an integrated environment, the payment received processes are the responsibility of the Enterprise Resource Planning (ERP) financial system.

For more information, see the *Financial Adapter Technical Overview Guide*.

## Recording payments received

Payments can be received by the organization from persons, employers, or unknown sources. Payments can also be received in bulk through batch processing. For example, the organization may have an agreement in place with various banks to receive electronic funds transfer (EFT) payments in bulk, and this is managed through a batch process.

- **Recording payments received from a person or an employer**

When a payment is received, the system creates a payment received instrument, instruction, and instruction line item to reflect the information entered about the payment.

The payment received instruction is added to the person or employer's list of financials.

The payment can then be allocated toward a liability that the person or employer holds with the organization.

- **Recording payments received in a suspense account**

When a payment is received from an unknown source, that payment is recorded in a suspense account until the organization can determine who the payment is from.

## Transferring payments received from a suspense account

Payments recorded in a suspense account can be transferred to a person or employer's list of financials.

A record of the payment received is maintained in the suspense account for accountability and traceability purposes. When transferred, the status of the payment received becomes "transferred" and a transfer date is recorded. After the transfer has been made, the received payment can be allocated toward a liability that the person or employer holds with the organization.

## Allocating payments received

A payment received can be allocated toward any number of outstanding liability instructions by the person or employer from whom the payment was received.

For example, if a person makes a payment of \$100 to the organization, that \$100 can be allocated toward one or more of that person's outstanding liabilities.

The amount of the allocation is deducted from the outstanding amount of the payment received instruction amount and applied toward the outstanding amount for the liability instruction. The allocation amount cannot be greater than the amount of the payment received. The allocation amount must also be less than or equal to the outstanding amount on the liability instruction, unless over allocation processing is set up on the liability.

To assist agency workers in allocating available funds towards outstanding liabilities, the unallocated amount on a payment received is tracked over time and displayed on the payment received instruction.

## Liability over allocation

A liability that is sent out by the organization can be an estimate only of what a participant might be billed. Therefore, a participant might send the organization more money than the billed amount. A liability product can be set up to allow for the amount that is sent in to be read as the correct amount to be allocated toward liabilities. This is called liability over allocation.

For example, an employer is issued a bill for service fees in the amount of \$100. That same employer sends in a payment for \$120. If over allocation functionality is set up for the liability, then the full \$120 can be allocated toward the liability line item. A new over allocation instruction line item is created in the amount of \$20. This \$20 remains as a credit reserved to the liability case. By reserving the amount, the original amount that is billed can be reconciled with the amount that might have been billed to determine whether the extra amount should be applied to extra liabilities on the case.

To allow this reserved amount to be applied toward extra liabilities on the case, the Reconcile Case Account batch process must be run. This batch process starts by unreserving the credit amount. It then looks for outstanding liabilities on the case and, if found, allocates the credit amount toward them. Any remaining credit amounts are made available for allocation to other liability cases the participant might have with the organization.

If over allocation functionality is not set up for a liability, the total allocations toward the liability cannot exceed the total amount for the liability. In the preceding example, only \$100 of the payment received can be allocated toward the liability instruction line item. If more than one allocation is made toward a liability line item, the total of all these allocations cannot exceed the total amount for the liability (\$100 in the preceding example).

Over allocation processing is set up for a liability product as part of financial administration.

## Refunding an unallocated amount

All or part of the unallocated amount of a payment that is received can be refunded to the person or employer from whom the payment was received. The amount is refunded through the preferred method of payment that is defined for the client.

For example, a payment of \$100 was received from a client and \$80 is later allocated toward an outstanding liability. However, if there are no further outstanding liabilities for the client, the remaining unallocated amount of \$20 can be issued back to the client. If the preferred payment method for the client is check, the system generates a check payment of \$20 for the client.

When the unallocated amount is refunded, the unallocated amount for the payment received instruction is updated and a refund instruction line item is created to issue the payment to the client. An allocation line item is also created to link the refund payment to the original payment received instruction.

A liability that is sent out by the organization might be an estimate only of what a participant is to be billed. A participant may therefore send the organization more money than the billed amount. A liability product can be set up to allow for the amount sent in to be read as the correct amount to be allocated toward liabilities. This is called liability over allocation.

For example, an employer is issued a bill for service fees in the amount of \$100. That same employer sends in a payment for \$120. If over allocation functionality is set up for the liability, then the full \$120 can be allocated toward the liability line item. A new over allocation instruction line item is created in the amount of \$20. This \$20 remains as a credit reserved to the liability case. By reserving the amount, the original amount that is billed can be reconciled with the amount that might have been billed to determine whether the extra amount should be applied to extra liabilities on the case.

To allow this reserved amount to be applied toward extra liabilities on the case, the Reconcile Case Account batch process must be run. This batch process starts by unreserving the credit amount. It then looks for outstanding liabilities on the case and, if found, allocates the credit amount toward them. Any remaining credit amounts are made available for allocation to other liability cases the participant may have with the organization.

If over allocation functionality is not set up for a liability, the total allocations toward the liability cannot exceed the total amount for the liability. In the preceding example, only \$100 of the payment that is received can be allocated toward the liability instruction line item. If more than one allocation is made toward a liability line item, the total of all these allocations cannot exceed the total amount for the liability, that is, \$100 in the preceding example.

Over allocation processing is set up for a liability product as part of financial administration.

## Canceling a refund

A refund of an unallocated amount that is already issued can be canceled.

For example, where a change of circumstance results in a further liability for the client the refund for the unallocated amount can be canceled. Another example is where the check that is issued to the client is lost in transit and it might be required that the agency cancel the refund payment.

When a refund payment is canceled, the unallocated amount of the payment received instruction is not updated. Instead, a replacement transaction is created that has an unallocated amount equal to the refund payment amount that was canceled. This is done for accountability and traceability purposes. The unallocated amount in the replacement transaction can then be allocated toward outstanding liabilities or if required it can be refunded to the client.

## Canceling payments received

A payment received can be canceled.

For example, if an employer's check bounces after it is recorded on the system, the payment received can be canceled to reflect that the check amount was not received by the organization.

When a payment received is canceled, the system creates a reversal instruction that represents the amount of the original payment received instruction.

If the payment received is allocated toward any outstanding liabilities, these allocations must also be canceled. For each allocation, an instruction line item is created with an amount equal to the original allocation amount. These instruction line items are liabilities that reflect the amount that were previously cleared by the allocations, but that is again owed to the organization as a result of the payment received cancellation.

If an unallocated amount of the payment received is refunded, these refund payments must also be canceled. Depending upon the value of the application property `curam.financial.createpaymentcorrectiononrefundcancel`, the system creates either an overpayment case or a payment correction case for the amount that is already refunded to the client. However, if the refund payment that is issued to the client is already canceled, the replacement transaction that was created as a result of canceling the refund payment is instead canceled.

For example, a payment of \$100 is received from the client and \$80 is allocated toward an outstanding liability. Later, the agency refunds the remaining unallocated amount of \$20 to the client through a check payment. If the payment for the \$100 received from James Smith bounced, the agency can cancel the entire payment received. When the agency cancels the payment that is received, any refunds that are associated with it must also be reversed and either an overpayment case or a payment correction case for \$20 must be created to offset the refund amount that is issued to James Smith. If the refund payment of \$20 is already reversed before the cancellation of the payment received; the replacement transaction that was created as a result of canceling the refund is instead canceled.

## 1.6 Financial instruction types

---

Financial processing uses different financial instruction types to process instruction line items.

### Payment instruction

The payment financial instruction type includes the benefit payments issued to eligible clients rolled up with any deductions, for example taxes.

#### Rolled-up instruction line items

The following table describes the financial instruction line items that can be rolled-up into a single payment instruction.

Table 3: Rolled-up financial instruction line items for payment instruction.

ILIs	Description	Created from process
Benefit ILI, for example, Max Personal, Child Allowance	A payment issued in relation to an eligible case component of type benefit.	Primary FC during Payment Generation
Deduction Item ILI	An amount deducted from a payment for an applied, un-applied, or third-party deduction.	Secondary FC during Payment Generation
Ad Hoc Bonus ILI	A once-off bonus payment that is directly related to the benefit. Ad hoc bonuses are rolled-up with the related benefit.	Ad Hoc Bonus FC during Payment Generation
Tax ILI	An amount subtracted at a set rate from a payment for tax purposes.	Process Tax Process during Payment Generation
Under Payment ILI	Changes in case circumstance can result in an eligible client being over or underpaid. Under payments can be automatically rolled-up in payment instructions thus increasing the payment instruction amount. Over payments cannot be automatically rolled-up into payment instructions. Automatic over payment processing for a benefit results in an overpayment case.	Reassessment Process

### Related instruction line items

The following table describes the financial instruction line items that can be related to the payment instruction.

Table 4: Financial instruction line items related to payment instruction.

ILIs	Description	Related through process
Reversal ILI	When a payment instruction is canceled, a reversal instruction line item is created to reverse the payment instruction amount. Any deductions and taxes applied to the original payment will also be canceled. If deductions were applied to the original payment instruction, two additional reversal instruction line items are created: one that cancels the deduction amount applied to the payment instruction and the other that cancels the deduction payment instruction line item (that may or may not have been issued to a third party depending on deduction processing).	Cancel Payment Instruction Process
Manual Payment ILI	During the capture manual payment process, a manual payment instruction line item is created to record the payment made. This instruction line item is compared to the actual payment amount due, and any differences in this comparison will result in the creation of an over or under payment.	Capture Manual Payment Process

## Liability instruction

The liability financial instruction type includes the bills that are issued to eligible clients rolled up with any surcharges as well as over payments, under payments, or both.

### Rolled-up instruction line items

The following table describes the financial instruction line items that can be rolled-up into a single liability instruction.

Table 5: Rolled-up financial instruction line items for liability instruction.

ILIs	Description	Created from
Liability ILI, for example Service Fee	A bill that is issued in relation to an eligible case component of type liability.	Primary FC during liability generation.
Surcharge ILI	An amount added to an overdue liability at a set rate.	Process surcharge process.
Over or Under Billing ILI	Changes in case circumstance can result in an eligible client being over or under-billed. Over and under billings can be automatically rolled-up in liability instructions. Over billings will decrease the amount due on a bill; under billings will increase the billing amount.	Reassessment process.

### Applied instruction line items

The following table describes the financial instruction line items that can be applied to the liability instruction.

Table 6: Financial instruction line items applied to liability instruction.

ILIs	Description	Applied as a result of process
Allocated Payment Received ILI	All or part of a payment received can be applied toward the entire liability instruction or toward individual liability instruction line items.	Allocate payment received process.
Overallocation ILI	When overallocation is enabled on a liability, it is possible to allocate a payment received amount that is over the amount due on the liability instruction. The liability reconciliation process reconciles the amount that is received against the original amount billed.	Allocate payment received process.
Write-Off ILI	A liability instruction can be written off in full or else all or part of individual liability instruction line items can be written off. Any written off amount reduces the amount that is owed on a bill.	Write-off liability process.
Reversal ILI	A liability instruction can be reversed in full or else all or part of individual liability instruction line items can be reversed. Any reversed amount reduces the amount owed on a bill.	Reverse liability process.

ILIs	Description	Applied as a result of process
Reverse Allocation ILI	When allocations have been made toward a liability and that liability is reversed, reverse allocation instruction line items are created. These instruction line items will make the allocation amount available to be applied toward future liabilities.	Reverse liability process.

## Payment received instruction

The payment received instruction type is used to generate a credit financial instruction from a payment received. Payments can be received from anonymous sources, from the client, or on the client's behalf.

### Rolled-up instruction line item

The following table describes the financial instruction line items that can be rolled-up into a single payment received instruction.

*Table 7: Rolled-up financial instruction line items for a payment received instruction.*

ILIs	Description	Created from
Payment Received ILI (Unallocated Payment Received)	An amount received by the organization from a person, employer, or unknown source that has yet to be allocated toward an outstanding liability.	Receive payment process.
Allocated Payment Received IL	All or part of a payment received that has been allocated toward an outstanding liability. Payments received can be allocated toward liabilities as part of receive payment process, thus getting rolled-up in initial payment received instruction.	Allocate payment received process.
Deduction Payment ILI	This is the amount equal to the third-party deduction applied to a benefit that is automatically rolled-up into a payment received instruction so that it can then be allocated toward outstanding liabilities.	Secondary FC during payment generation.
Refund ILI	All or part of the unallocated amount of the payment received that has been issued to the client. If a refund payment has been issued, it is reflected on the payment received instruction.	Refund unallocated amount process.

### Applied instruction line items

The following table describes the financial instruction line items that can be applied to the payment received instruction.

Table 8: Financial instruction line items applied to a payment received instruction.

ILIs	Description	Applied as a result of process
Reverse Allocation ILI	When reversing a payment received, any allocations that are made toward the liability from the payment received will be checked and reverse allocation instruction line items created for each which will make the amount due again on the liability.	Reverse allocated payment received process.
Reversal ILI	A payment received can be canceled, making the funds no longer available for allocation toward outstanding liabilities. As described, reverse allocation instruction line items are created, if applicable.	Reverse Payment received instruction process.

## Reversal instruction

The reversal instruction type is used to negate all or part of an existing financial instruction. A reversal can either be a credit or a debit depending on the type of existing financial instruction being reversed. When reversing a liability instruction, or liability instruction line items, or when reversing a payment instruction through payment cancellation, the reversal is a credit. When reversing a payment received through payment received cancellation, the reversal is a debit.

### Rolled-up instruction line item

The following table describes the financial instruction line item that is rolled-up into a reversal instruction.

Table 9: Rolled-up financial instruction line item for reversal instruction.

ILIs	Description	Created from
Reversal ILI	For reversed liabilities and payments, the reversal instruction line item is a credit. For canceled payments received, the reversal instruction line item is a debit.	Reverse liability ILI process or cancel payment instruction process or cancel payment received instruction process.

### Related instruction line items

The following table describes the financial instruction line items that can be related to the reversal instruction.

Table 10: Financial instruction line items related to reversal instruction.

ILIs	Description	Related through process
Benefit ILI, for example, max personal, child allowance	When a payment is canceled, all benefit instruction line items are rolled-up into the reversal instruction.	Payment instruction creation process.

ILIs	Description	Related through process
Additional ILIs rolled-up in original payment instruction	When a payment is canceled, all additional instruction line items rolled-up in the original payment instruction must be accounted for. These include deduction, ad hoc, tax, and under payment instruction line items.	Payment instruction creation process.
Liability ILI	Any or all liability instruction line items that have been reversed are rolled-up into the reversal instruction.	Liability instruction creation process.
Over or under billing ILI	When a liability is reversed in full, any over or under billings rolled-up in the liability must be accounted for.	Reassessment process.
Payment received ILI (unallocated payment received)	When a payment received is canceled, all unallocated payment received instruction line items are rolled-up into the reversal instruction.	Payment received creation process.
Allocated payment received ILI	When a payment received is canceled, all allocated payment received instruction line items are rolled-up into the reversal instruction.	Allocate payment received process.
Reverse allocation ILI	When allocations have been made toward a liability and that liability is reversed, reverse allocation instruction line items are created. These instruction line items will make the allocation amount available to be applied toward future liabilities. When reversing a payment received, any allocations that are made toward the liability from the payment received will be checked and reverse allocation instruction line items created for each which will make the amount due again on the liability.	Reverse liability process (when allocations have been made toward the liability) or reverse allocated payment received process.

### Applied instruction line item

The following table describes the financial instruction line item that can be applied to the reversal instruction.

Table 11: Financial instruction line item applied to reversal instruction.

ILIs	Description	Applied as a result of the process
Write-Off ILI	When all or a portion of a payment received is applied to a liability, and then that payment received is reversed, the amount that is applied to the liability is reversed via the creation of reverse allocation instruction line items. All or part of the reverse allocation line items can be written-off, resulting in the creation of write-off instruction line items.	Write-off reversed allocated payment received ILI process.

## Write-off instruction

A write-off instruction is a credit transaction to undo all or part of an existing liability instruction or all or part of a reversed payment received instruction.

### Rolled-up instruction line item

The following table describes the financial instruction line item that is rolled-up into a write-off instruction.

Table 12: Rolled-up financial instruction line item for write-off instruction.

ILIs	Description	Related through process
Write-off ILI	Write-off instruction line items are created for all or part of the liability instruction, liability instruction line items, or reversed allocation payment received instruction line items and rolled-up into the write-off instruction.	Write-off liability ILI process or write-off reversed allocated payment received ILI process.

### Related instruction line items

The following table describes the financial instruction line items that can be related to the write-off instruction.

Table 13: Financial instruction line items related to write off instruction.

ILIs	Description	Related through process
Liability ILI, for example, service fee	The liability instruction that includes written-off instruction line item or items is rolled-up in the write-off instruction.	Liability instruction creation process.
Reverse allocation ILI	The reverse allocation instruction line items for the written-off, reversed liability, or reversed payment received are rolled-up in the write-off instruction.	Reverse liability process (when allocations have been made toward the liability) or reverse allocated payment received process.

## Third-party payment instruction

A third-party payment instruction is issued to a third party in relation to third-party deductions or tax deductions that are applied to benefits.

The following table describes the financial instruction line item that is rolled-up into a third-party payment instruction.

Table 14: Rolled-up financial instruction line item for third payment instruction.

ILIs	Description	Created from
Deduction payment ILI	The issue concern payments batch process searches for all unprocessed third-party payment instruction line items for a participant and rolls them up into one payment instruction.	Issue concern payments batch process.
Tax payment ILI	The amount equal to the taxes applied to a benefit that can then be rolled-up into a third-party payment instruction for the relevant tax authority.	Issue concern payments batch process.

## Adjustment instruction

An adjustment instruction is created to correct a client's financial account.

The following table describes the financial instruction line item that is rolled-up into an adjustment instruction.

Table 15: Rolled-up financial instruction line item for adjustment instruction.

ILIs	Description	Created from
Adjustment ILI	Adjustment amount to be applied to client's financial account and whether that amount is a credit or a debit.	Create adjustment process.

## 1.7 Financial batch processes

Financial batch processes are used to create and maintain instruction line items and their associated financial instructions.

### Submitting to the batch queue

Each batch process to be executed must first be submitted to the batch queue via the System Administrator application.

This is done by selecting the *Execute* action for that batch process from the list. If parameters are required, the user is prompted to enter them on the Execute Batch Process page. The parameters vary depending on the batch process and some are optional. Once the user enters the parameter value(s), they should then click the *Execute* button to submit the process to the queue. For batch processes that do not require any parameters, the user must confirm the execution of the process. Additionally, the user has the option of canceling at this point.

On submission to the queue, the following entities are populated, BatchParamValue only being populated if parameters are required:

- BatchProcRequest
- BatchParamValue

One important thing to note about submitting processes to the queue is that they should be submitted in the order in which they must run, so if process A must run before process B, process A must be executed first. This is particularly important when creating financials.

## Running the batch launcher

Processes that are submitted to the queue are picked up by the batch launcher. The batch launcher is part of the Server Development Environment (SDEJ).

When the batch launcher is run, the first thing it does is start the stand alone keyserver - this will be required if any batch process is performing inserts onto the database.

When all the batch processes have executed, the batch launcher will stop the stand alone keyserver.

Note that batch process streaming allows multiple instances of a single batch process to be run in parallel, making full use of the available processing.

For more information about batch process streaming, see the *Batch Streaming Developers Guide*.

## Running a batch program from the command line

Run a command from the main project directory, *EJBServer*.

The following command can be run from the main project directory, *EJBServer*, substituting the appropriate values for <username>, <ClassName> and <OperationName> as well as any parameter name-value pairs:

```
build runbatch
-Dbatch.username=<username>
-Dbatch.program=curam.core.intf.<ClassName>.<OperationName>
-Dbatch.params="param1=param1value, param2=param2value"
```

## Financials batch suite

To create the financial records, run four batch processes in sequence.

In order for the financials records to be created, the following batch processes must be run in this sequence:

1. DetermineProductDeliveryEligibility
  - Parameters - instanceID, processingDate and productID
  - Processes - Cases to FCs
  - Pre-run Status - Approved ('CS6')
  - Post-run Status - Active ('CS1') if Eligible, PendingClosure ('CS7') if Ineligible, Suspended ('CS2') if error occurred
  - Summary of Processing - Identifies cases with status of Approved, Activates and assesses case eligibility, and generates FCs
2. GenerateInstructionLineItems

- Parameters - `deliveryMethod`, `instanceID`, `processingDate`, `processingDateFrom`, `processingDateTo` and `productID`
  - Processes - FCs to ILIs
  - Pre-run Status - Live ('LIV')
  - Post-run Status - Live ('LIV') if the FC is not expired; or Closed ('CLD') if `nextProcessingDate > processingDate`
  - Summary of Processing - Reassesses each case being processed, generates ILIs for live FCs and rolls forward their `nextProcessingDate`. Expires FCs if `nextProcessingDate` is after processing date
3. `GenerateInstruments`; and optionally
- Parameters - `instanceID` and `processingDate`
  - Processes - ILIs to Instructions and Instruments
  - Pre-run Status - Unprocessed ('UNP')
  - Post-run Status - Processed ('PRO')
  - Summary of Processing - Picks up unprocessed ILIs and rolls them into instructions and instruments
4. `GeneratePayslips`
- Parameters - `processingDate`
  - Processes - Payslips and PayslipInstructions
  - Pre-run Status - Pending Issue ('PS2') or Created ('PS1')
  - Post-run Status - Issued ('PS3')
  - Summary of Processing - Picks up Payslips pending issue and generates the Payslip for them

## Business processing date

The business processing date is an optional parameter that is common to all batch jobs.

The parameter is distinct from the system date. During batch execution the system date is always 'today' but the business processing date can be specified using the `ProcessingDate` parameter of the particular batch job. This gives us the ability to 'spooft' the date that a batch process was run. For example, if a financial batch run was scheduled to run on a Friday but failed. The organization could re-run it on Saturday but set the `ProcessingDate` to Friday's date. This means that all the calculations etc. would behave as though it was genuinely being run on the Friday. If the `ProcessingDate` parameter is not specified the business processing date defaults to the system date.

At a technical level, within any transactions initiated by the batch process, calls to `getCurrentDate()` will return the business processing date specified as the batch parameter. Any calls to `getSystemDate()` will return the system date, that is, 'today', as normal.

## Output to logs, emails, and reports

When a batch process is being designed, the developer will usually want some sort of output containing summary information, such as the number of records processed, the execution time

and so on. If the information is written to a log file, this file is located in the *EJBServer/buildlogs* directory.

If 'from' and 'to' email addresses were specified during the installation process then an email, containing the same information as the log file above, will be sent to the 'to' email address.

Any reports from the batch process can be found in the same location as the log and will have a .dat file extension.

## GenerateInstructionLineItems

`GenerateInstructionLineItems` is the batch process that produces Instruction Line Items (ILIs) from financial components (FCs). It identifies each financial component of a particular delivery method that reaches its next processing date within the dates specified.

`GenerateInstructionLineItems` is run directly after the `DetermineProductDeliveryEligibility` batch process, or at any time after a case is activated.

`GenerateInstructionLineItems` is run to generate Instruction Line Items that are eventually rolled up by the `GenerateInstruments` batch process to produce payment instructions or instruments and liability instructions or instruments.

### How does GenerateInstructionLineItems work?

The following list outlines the parameters that the batch process looks for when the batch process is submitted to the queue from the application:

1. `DeliveryMethod`.
2. `InstanceID`.
3. `ProcessingDate`.
4. `ProcessingDateFrom`.
5. `ProcessingDateTo`.
6. `ProductID`.

**Note:** These parameters are not mandatory. If the dates are not entered, the dates default to the system null date. If `DeliveryMethod` or `ProductID` is not specified, all FCs are processed regardless of the delivery method and product. Ensure that the `InstanceID` is specified when you use the batch streaming architecture.

When the batch process runs, it determines the FCs to process based on the specified parameters.

- **DeliveryMethod**

The following list outlines the FCs that are processed if you do not specify the `DeliveryMethod`:

- Liability FCs.
- Payment FCs.
- Recoupment FCs.

- Liability FCs, that is, FCs of category LBY are processed for the delivery method of Invoice.
- Payment FCs, that is, FCs of category CLM are processed for each delivery method that is listed in the `MethodOfDelivery` code table. Examples include Cash (CSH), Check (CHQ) and EFT (EFT).
- Recoupment FCs, that is, FCs of category RCP are processed for each delivery method that is listed in the `LibMethodOfDelivery` code table. Examples include Giro (GIR) and Invoice (INV).

**Note:** If the `DeliveryMethod` is specified as INV, that is, Invoice, then only Liability FCs are processed. Otherwise, Payment and Recoupment FCs are processed for the specified `DeliveryMethod` as, for example, Cash (CSH), Check (CHQ), Giro (GIR).

- **InstanceID**  
When you run the batch streaming architecture, ensure you specify the InstanceID.
- **ProcessingDate**  
When you run the batch process, ProcessingDate is the business date to use. For more information, see the *Business processing date* related link.
- **ProcessingDateFrom**  
If ProcessingDateFrom is not specified when you submit the batch process, it defaults to the system null date.  
If ProcessingDateFrom was incorrectly set to a future date, it defaults to the system null date when the batch process runs. Otherwise, it uses the specified date.
- **ProcessingDateTo**  
ProcessingDateTo defaults to the business processing date inside the batch process if the parameter defaulted to a null date on submission, that is, no date was specified. The reason ProcessingDateTo is not defaulted to the current date on submission to the queue if not specified because the client current date might not be synchronized with the server current date. In general terms, this can potentially lead to problems when the batch process runs.  
If ProcessingDateTo was set incorrectly to a future date, it defaults to the business processing date when the batch process runs. Otherwise, the date that is specified is used.
- **ProductID**  
If ProductID was not specified, cases for all products are processed.

### What can I configure or customize?

You can configure the system to control whether Cúram Express Rules (CER) cases are reassessed before you generate payments. The following steps outline how to configure the system in this way:

1. Log in as a system administrative user.
2. Click **Application > Miscellaneous**.
3. Set the `ENV_GENERATEINSTRUCTIONLINEITEMS_DONT_REASSESS_CASE` application property to **YES**. The default value is **NO**.
4. To publish the property change, click **Publish**.

**Related concepts**

[Business processing date on page 50](#)

The business processing date is an optional parameter that is common to all batch jobs.

**GenerateInstruments**

`GenerateInstruments` is the batch process that identifies unprocessed instruction line items (ILIs), that is, instruction line items with a status of UNP, and processes each one.

`GenerateInstruments` is run directly after the `GenerateInstructionLineItems` batch process.

`GenerateInstruments` is run to generate the payment instructions or instruments and liability instructions or instruments that create the respective payments and bills that are issued to a participant.

**How does GenerateInstruments work?**

The following list outlines the parameters the batch process looks for when the batch process is submitted to the queue from the application:

1. InstanceID.
2. ProcessingDate.

**Note:** The parameters are not mandatory.

- **InstanceID**

When you run the batch streaming architecture, ensure you specify the InstanceID.

- **ProcessingDate**

When you run the batch process, ProcessingDate is the business date to use. For more information, see the *Business processing date* related link.

The following list outlines the order that ILIs are processed when the batch job runs:

- Product delivery ILIs due, that is, all payment and liability Instruction Line Items.
- Tax ILIs due, that is, rolls up Instruction Line Items of category TAX to issue payments to the tax authority.
- Applied deduction ILIs due, that is, Instruction Line Items of category DED that are applied to a liability.
- Repayment ILIs due, that is, Instruction Line Items of category PRV.
- Third-party deduction ILIs due, that is, rolls up Instruction Line Items of category DED to issue payments to third parties.

**Related concepts**

[Business processing date on page 50](#)

The business processing date is an optional parameter that is common to all batch jobs.

## **GeneratePayslips**

`GeneratePayslips` is the batch process that identifies payslips that are pending issue. `GeneratePayslips` is run directly after the `GenerateInstruments` batch process.

### **Why is GeneratePayslips run?**

`GeneratePayslips` is run to generate the payslips that are sent to the participants to reflect the breakdown of their payments or bills. The breakdown is given at the Instruction Line Item (ILI) level. The following list outlines the different types of payslip:

- Client.
- Case nominee.
- Participant.
- Third party.

### **How does GeneratePayslips work?**

When the batch process is submitted to the queue from the application, it looks for the parameter `ProcessingDate`. The parameter `ProcessingDate` is not mandatory.

Use the business date `ProcessingDate` when you run the batch process. For more information, see the *Business processing date* related link.

When the batch job runs, it processes payslips with a status of pending issue (PS2). As it processes the payslips, the batch job decides what type of payslip it is based on its components' recipient type. The batch job opens an output file for the payslip type and writes all payslips of this type into this file. Output data files for a particular type are opened only after each run of the batch process.

### **Related concepts**

[Business processing date on page 50](#)

The business processing date is an optional parameter that is common to all batch jobs.

## **LoadPaymentsReceived**

`LoadPaymentsReceived` is the batch process that loads payment received details from an external flat file onto the system for persistent storage.

When the batch process runs, it opens an input file and processes each record listed in the file. The data in this input file is provided in tab delimited format.

The batch process is run whenever the input file becomes available to the organization, for example, from a bank, to reflect payments that have already been received. The data needs to be entered onto the system to reflect these payments.

### **How does LoadPaymentsReceived work?**

When the batch process is submitted to the queue from the application, it looks for the parameters `FilePath`, `FileName`, and `ProcessingDate`.

The parameters `FilePath` and `FileName` are mandatory. The parameter `FilePath` is typically in the form `<Drive>:/Curam/svr/run`, that is, the directory where the input file resides.

The parameter `FileName` is the full name of the input file, including extension, that contains details of the payments received.

Use the `ProcessingDate` business date while executing the batch process. For more information, see the *Business processing date* related link. When run, the batch process looks for the `FileName` specified in the `FilePath` specified.

**Note:** When the `curam.participant.enableibanfunctionality` application property is set to **True**, then the batch input file also requires IBAN and BIC values.

### Related concepts

[Business processing date on page 50](#)

The business processing date is an optional parameter that is common to all batch jobs.

## IssueConcernPayments

`IssueConcernPayments` is the batch process that issues payments to persons, employers, external parties, information providers, service suppliers, product providers, utilities, and representatives.

`IssueConcernPayments` identifies persons, employers, external parties, information providers, service suppliers, product providers, utilities, and representatives who are due to be paid and issues their respective payments. The payment method and frequency for each of the participant types are held on the respective entities and are set during the registration process.

The times at which the process is run depends on the next processing dates of the relevant participants. It is conceivable that the process be run daily, as the next processing dates of the participants, that is, persons, employers, external parties, information providers, service suppliers, product providers, utilities, and representatives, might cover the entire week for all types. Typically, the dates are held on the organization calendar.

### How does IssueConcernPayments work?

The following list outlines the parameters that the batch process looks for when the batch process is submitted to the queue from the application:

1. `ConcernTypeCode`.
2. `MethodOfPayment`.
3. `NextPaymentDateFrom`.
4. `NextPaymentDateTo`.
5. `ProcessingDate`.

**Note:** The parameters are not mandatory. If the dates are not entered, the dates default to the system null date.

- **ConcernTypeCode**

If `ConcernTypeCode` is not specified, all concern role types that are listed in the `ConcernRoleType` code table are processed. The parameter applies to persons, employers, external parties, information providers, service suppliers, product providers, utilities, and representatives. Nothing is picked up for prospect person or prospect employer.

All records of the specified type are processed even if the type does not exist, that is, the batch process never fails if an incorrect type is specified. Payments are issued only for participants of type person (RL1), employer (RL2), external party (RL17), information provider (RL5), service supplier (RL3), product provider (RL4), utility (RL6), or representative (RL13).

- **MethodOfPayment**

If `MethodOfPayment` is not specified, all delivery methods that are listed in the `MethodOfDelivery` code table are processed. Otherwise, processing is performed for the method of payment that is specified only.

- **NextPaymentDateFrom and NextPaymentDateTo**

If `NextPaymentDateFrom` or `NextPaymentDateTo` is not specified when submitting the batch process, they default to the system null date.

`NextPaymentDateFrom` and `NextPaymentDateTo` are not defaulted to the current date on submission to the queue because if the user does not specify them the client date might not be synchronized with the server current date. In general terms, this can potentially lead to problems when batch processes run.

Inside the batch process, two processing date parameters, `dateFrom` and `dateTo`, are set based on the values of `NextPaymentDateFrom` and `NextPaymentDateTo`. The following list outlines the checks that are performed:

- If both `NextPaymentDateFrom` and `NextPaymentDateTo` are null dates, that is, not specified, `dateFrom` and `dateTo` are set to the current system date.
- If `NextPaymentDateFrom` was specified and `NextPaymentDateTo` was not, both `dateFrom` and `dateTo` are set to the `NextPaymentDateFrom` value.
- Similarly, if `NextPaymentDateTo` was specified and `NextPaymentDateFrom` was not, both `dateFrom` and `dateTo` are set to the `NextPaymentDateTo` value.
- If `NextPaymentDateFrom` and `NextPaymentDateTo` are specified, `dateFrom` is set to the `NextPaymentDateFrom` value with `dateTo` being set to `NextPaymentDateTo`.

- **ProcessingDate**

When you run the batch process, use the business date `ProcessingDate`. For more information, see the *Business processing date* related link.

### Related concepts

[Business processing date on page 50](#)

The business processing date is an optional parameter that is common to all batch jobs.

## ExpirePayments

`ExpirePayments` is the batch process that expires payments that are not cashed after a certain length of time.

`ExpirePayments` identifies payment instruments on the system that have a `reconcilStatusCode` of 'issued' (ISS) and expires them if they are on the system for a certain length of time.

The purpose of `ExpirePayments` is to expire payments of a specified delivery method. Typically, these are of type check (CHQ) that were not cashed after some days. Checks usually have a six-month lifespan and cannot be cashed after this. The organization wants to run the batch process to expire the necessary payments.

### How does `ExpirePayments` work?

The following list outlines the parameters that the batch process looks for when the batch process is submitted to the queue from the application:

- `DeliveryMethod`.
- `ExpiryPeriod`.
- `ProcessingDate`.

Enter the `ExpiryPeriod` in days, that is, the number of days the payments must be on the system to be picked up by the batch process. The `ProcessingDate` is the business date to use when you run the batch process. For more information, see the *Business processing date* related link.

### Related concepts

[Business processing date on page 50](#)

The business processing date is an optional parameter that is common to all batch jobs.

## ProcessPaymentInstrumentTypes

`ProcessPaymentInstrumentTypes` is the batch job that processes all payment instrument records that are due for issue and writes their details into an output file.

`ProcessPaymentInstrumentTypes` identifies the payment instruments that are to be issued, populates an output file with these instruments' details and updates their status to issued (ISS). Running this program and creating the output file is the equivalent of issuing payments. For example, if the batch process is run for `DeliveryMethodType` of EFT, the output file is sent to the bank where the payments are transferred to the participant accounts.

Run `ProcessPaymentInstrumentTypes` to provide a list of payments due to the financial institution or institutions that provide the payments to the participant or participants.

### How does `ProcessPaymentInstrumentTypes` work?

The following list outlines the parameters that the batch process looks for when the batch process is submitted to the queue from the application:

- `DeliveryMethodType`.
- `ProcessingDate`.

The parameters are not mandatory. If the delivery method type is not provided, all delivery method types are processed. Otherwise, the one provided is processed. Use the `ProcessingDate` is the business date to use when you run the batch process. For more information, see the *Business processing date* related link.

### **Related concepts**

[Business processing date on page 50](#)

The business processing date is an optional parameter that is common to all batch jobs.

## **Payment reconciliation**

Payment reconciliation is the batch process that reconciles an account by comparing what was due to be paid with the amount paid, reporting any discrepancies.

When the batch process runs, it compares the intended amount to be paid to a participant against the amount paid. The input file to the process contains details of the payments that are received by the participants. The file comes from the institution that made the payments, for example, a bank. The payments in this flat file are compared to the payments issued by the system. Any discrepancies found are generated in a report.

Run the payment reconciliation batch to identify discrepancies, if there are any, in amounts that are paid against the intended amounts to pay.

### **How does payment reconciliation work?**

When the batch process is submitted to the queue from the application, it looks for the parameters `FilePath`, `FileName`, and `ProcessingDate`.

The parameters `FilePath` and `FileName` are mandatory. The parameter `FilePath` is typically in the form `<Drive>:/Curam/svr/run`, that is, the directory where the input file resides.

The parameter `FileName` is the full name of the input file, including extension, that contains details of the payments received.

Use the `ProcessingDate` business date while you run the batch process. For more information, see the *Business processing date* related link. When run, the batch process looks for the `FileName` specified in the `FilePath` specified.

### **Related concepts**

[Business processing date on page 50](#)

The business processing date is an optional parameter that is common to all batch jobs.

## **GeneralLedgerInterface**

`GeneralLedgerInterface` is the batch process that gathers financial transactions for a specified date, or date range, and exports them from the application into an output file that can then be imported into the general ledger.

The output file contains details of financial transactions at the lowest level of granularity. In the application, financial transactions at this level are defined as instruction line items. The output file is a sample of what is sent to a third party who then uses the file to update the general ledger.

The `GeneralLedgerInterface` is run whenever the organization wants to create a financial transaction output file that can be imported into the general ledger.

The `GeneralLedgerInterface` is run to provide an output file of credit and debit transactions for a specified date, or date range, that can be imported into a general ledger interface.

### How does `GeneralLedgerInterface` work?

The following list outlines the parameters that the batch process looks for when the batch process is submitted to the queue from the application:

1. `CreationDateSearchInd`.
2. `DateFrom`.
3. `DateTo`.
4. `ProcessingDate`.

**Note:** The parameters are not mandatory. If the dates are not entered, they default to the system null date.

- **CreationDateSearchInd**

`CreationDateSearchInd` indicates whether the extract is based on the creation date or effective date. If `CreationDateSearchInd` was set to `True` by the user, a creation date range search is performed. Otherwise, an effective date range search is used.

When specifying the value for `CreationDateRangeInd`, it must be either `True` or `False`. It cannot be 1 or 0, or `Yes` or `No`. Validation is performed on the client to prevent entries of this type.

- **DateFrom and DateTo**

The user does not specify `DateFrom` or `DateTo` when submitting the batch process. The parameters default to the system null date.

`DateFrom` and `DateTo` are not defaulted to the current date on submission to the queue if the user does not specify them because the client current date might not be synchronized with the server current date. In general terms, this can potentially lead to problems when batch processes run.

The following list outlines the date checks that are performed inside the batch process:

- If `DateFrom` is found to be a null date, that is, not specified by a user, produces an error and the batch process fails.
- Similarly, if `DateTo` is found to be a null date, that is, not specified by a user, produces an error and the batch process fails.

- **ProcessingDate**

Use the `ProcessingDate` business date while you run the batch process. For more information, see the *Business processing date* related link.

### Related concepts

[Business processing date on page 50](#)

The business processing date is an optional parameter that is common to all batch jobs.

## ReconcileCaseAccount

`ReconcileCaseAccount` is run to reconcile liability cases where a liability overpayment or liability underpayment occurs. It works only for liability over billing and liability under billing. It does not reconcile benefit overpayment or payment corrections.

Use `ReconcileCaseAccount` to balance a single case account. During the lifetime of a liability case, liability transactions might be added to the case. Over allocation might occur due to one or more of these for reserving the money for this case. So, the reconcile case batch process does take the over allocations and see whether they can be allocated to another liability transactions on the same case. When all liabilities are paid, if any money remains it is made available for allocation to other cases. However, the allocation to other cases is not automated.

The following list outlines the batch process performs two separates pieces of processing:

- It reconciles all liability cases where an underpayment is applied.
- It also reconciles all liability cases where an overpayment is applied.

Run the `ReconcileCaseAccount` to reconcile both overpayment and underpayment liability cases.

### How does ReconcileCaseAccount work?

When the batch process is submitted to the queue from the application, it looks for the `ProcessingDate`.

**Note:** The parameter is not mandatory.

- **ProcessingDate**

Use the `ProcessingDate` business date while you run the batch process. For more information, see the *Business processing date* related link.

The process reconciles both overpayment and underpayment liability cases. The following list outlines the order in which liability cases are performed:

1. Overpayment liability cases.
2. Underpayment liability cases.

The processing for reconciling overpaid liability cases retrieves the list of instruction line items for the overpaid liability case. If the instruction line items represent an overpayment and have a status of processed, the overallocation is converted into an unallocated payment received record. If the case the overpayment relates to has any outstanding liabilities, then the newly created payment received instruction is allocated against them.

The processing for reconciling underpaid liability cases retrieves the list of instruction line items for the underpaid liability case. If the instruction line items represent an underpayment and have a status of processed, an adjustment line item record is created that is allocated against outstanding liability instruction line items.

### Related concepts

[Business processing date on page 50](#)

The business processing date is an optional parameter that is common to all batch jobs.

## 1.8 Financial code tables

An overview of the financial code tables, including the code tables that you must customize to allow a product to issue financials. The Instruction Line Items (ILI) relationship types code table is used internally by the financial manager to process certain types of instruction line items. The overview includes financial code tables that you must customize to allow your product to issue financial and describes the code table that is used internally to the financial manager to link related instruction line items.

### Financial code tables

You must customize the five code tables to allow your product to issue financials. The examples use different values for the various code table entries to help explain how they are related, for example MY\_RCT and MY\_FCT. However, for simplicity you might consider using the same value for all of them, for example MY\_COMPx.

If the code tables are not customized, the `TypeCode` attribute of the `FinancialComponent` entity and, then, the `InstructionLineItemType` attribute of the `InstructionLineItem` entity, is blank.

- **RulesComponentType**

The `RulesComponentType` code table lists all the component types that are used in the rules. You must add an entry to this code table to represent your component and reference this entry in the `name` attribute of your custom Objective Type rule class. The following code shows an example:

```
<code>
  default="false" java_identifier=" "
status="ENABLED" value="MY_RCT"
  >
  <locale language="en" sort_order="0">
    <description>My rules component type</description>
    <annotation/>
  </locale>
</code>
```

- **FinComponentType**

The `FinComponentType` code table lists all the financial component types that can be generated by the Financial Manager. You must add an entry to this code table to represent the financial schedule for your component. The following code shows an example:

```
<code>
  default="false" java_identifier=" "
status="ENABLED" value="MY_FCT"
  >
  <locale language="en" sort_order="0">
    <description>My financial component type</description>
    <annotation/>
  </locale>
</code>
```

- **ProductComponentFCCConv**

The `ProductComponentFCCConv` code table maps the `RulesComponentType` codes that are used by the rules to the `FinComponentType` codes used by the Financial Manager. You must add an entry to the code table to map the custom codes for your component. Ensure that the `value` matches the entry you added to the `RulesComponentType` code table. Ensure that the `description` matches the entry you added to the `FinComponentType` code table. The following code shows an example:

```
<code
  default="false" java_identifier=" "
status="ENABLED" value="MY_RCT"
  >
  <locale language="en" sort_order="0">
    <description>MY_FCT</description>
    <annotation/>
  </locale>
</code>
```

- **ILIType**

The code table lists all of the instruction line item types that can be generated by the Financial Manager. You must add an entry to this code table to represent a financial transaction for your component. Ensure that the `value` matches the entry you added to the `FinComponentType` code table. The following code shows an example:

```
<code
  default="false" java_identifier=" "
status="ENABLED" value="MY_FCT"
  >
  <locale language="en" sort_order="0">
    <description>My ILI type</description>
    <annotation/>
  </locale>
</code>
```

- **TranslateILIType**

The code table maps the `ILIType` codes that are used to represent a financial transaction to the `ReassessmentAmount` codes used by the eligibility and entitlement engine during reassessment. You must add an entry to this code table to map the custom codes for your financial transactions (ILIs). Ensure that the `value` matches the entry from the `ILIType` code table. Ensure the `description` matches an entry from the `ReassessmentAmount` code table. Map custom ILIs that are considered payments to the `ReassessmentAmount` code AT1. Map custom ILIs that are considered liabilities to the `ReassessmentAmount` code AT6. The following code shows an example:

```
<code
  default="false" java_identifier=" "
status="ENABLED" value="MY_FCT"
  >
  <locale language="en" sort_order="0">
    <description>AT1</description>
    <annotation>My ILI Type -> Gross Amount</annotation>
  </locale>
</code>
```

## Instruction Line Items (ILI) relationship types

It is not necessary to customize the code table to allow your product to issue financials, but it is used internally by the financial manager when the financial manager is processing certain types of instruction line items (ILI).

An example of an ILI relationship is the relationship between a deduction item ILI and the corresponding deduction payment ILI. The relationship between ILIs is held on the instruction line item relation entity.

The main data that is held on this entity is the unique identifiers of the related ILIs and their relationship type. The following table lists all possible relationship types.

*Table 16: ILI relationship types.*

ILI relationship type	Description
REV	Reversals.
SUR	Surcharges.
TAD	Tax deduction.
TAP	Tax payment.
UTD	Utility deduction.
UTP	Utility payment.
RPP	Debt or overpayment payment.
RPD	Debt or overpayment deduction.
INT	Interest.
LLB	Loan liability.
LRP	Loan repayment.
CDI	Case deduction item.
CDP	Case deduction payment.
CAN	Cancellation.
RGN	Regeneration
ARV	Allocation reversal

## 1.9 Financial customization points

You can extend financial functions by using the customization points for financials in Cúram. You can enable customizations by adding a product type to the `curam.financial.alternativeimpl.producttypes` application property.

### Customization application property

The customization capability is enabled by adding a product type to the application property `curam.financial.alternativeimpl.producttypes`. More than one product type can be added by using a comma-delimited list. The product types that are listed must be valid codes from the `ProductType` code table.

If this property is found to contain the product type specific to the case currently being processed by the financial manager, the system looks for an alternative implementation of the relevant financial class by using the additional application properties.

By default these application properties are all blank and the core class implementations are used.

### Defining a custom implementation

The table lists the application properties that must contain the fully qualified name of the appropriate financial subclass.

Table 17: A list of application properties that must contain the fully qualified name of the appropriate financial subclass.

Application property	Description
<code>curam.financial.financialhook.createcancellation</code>	The property must contain a subclass of <code>curam.core.impl.CreateCancellation</code> .
<code>curam.financial.financialhook.casereassessment</code>	The property must contain a subclass of <code>curam.core.impl.CaseReassessment</code> .
<code>curam.financial.financialhook.createreversal</code>	The property must contain a subclass of subclass of <code>curam.core.impl.CreateReversal</code> .
<code>curam.core.hook.impl.PaymentInstructionLineItem</code>	The method can suspend the payment based on the maximum amount that is configured on the product. The method provides a hook for customers to override the default implementation by providing logic on their implementation class and bind the implementation class to the <code>PaymentInstructionLineItem</code> interface by using Guice.
<code>uram.core.impl.FinancialManagerHooks.setPaymentEffectiveDate</code>	The method can change the effective date for a payment instrument. The default implementation can be overridden by binding a customer's class to the <code>FinancialManagerHooks</code> interface by using Guice. The default implementation returns the effective date unchanged.

If any of the application properties are blank, the core class implementation is used.

## Financial hooks

Use the following financial hooks to extend financial functions and processing.

### **Deduction hooks**

Use the available hooks on the `curam.core.impl.FinancialHooks` abstract class to offset deductions against standalone underpayments, that is, underpayments that financials processing apply to the original benefit case.

### **About this task**

By default, an objective that is associated with an underpayment financial component on the original benefit case is not deductible. You can specify that a standalone underpayment objective is deductible by providing an implementation of the `curam.core.impl.FinancialHooks.isComponentDeductible()` customization point. You can also instruct financials processing to apply a specified case deduction to a standalone underpayment by providing an implementation of the `curam.core.impl.FinancialHooks.deductFromStandaloneUnderpayment()` customization point. For more information, see the Javadoc for the associated class.

### **Procedure**

1. Provide an implementation of the `FinancialHooks` class.

The following code sample shows how to implement the `FinancialHooks` class:

```
import curam.core.impl.FinancialHooks;
import curam.core.struct.CaseDeductionItemDtls;
import curam.core.struct.CaseID;
import curam.core.struct.DeductibleInd;
import curam.core.struct.RulesObjectiveID;
import curam.util.exception.AppException;
import curam.util.exception.InformationalException;

public class CustomFinancialHooks extends FinancialHooks {

    @Override
    public DeductibleInd isComponentDeductible(final CaseID caseID,
        final RulesObjectiveID rulesObjectiveID)
        throws AppException, InformationalException {

        // Add custom code...
    }

    @Override
    public boolean deductFromStandaloneUnderpayment(
        final CaseDeductionItemDtls dtls)
        throws AppException, InformationalException {

        // Add custom code...
    }
}
```

2. Bind the implementation in a Guice module.

The following code sample shows how to bind the implementation:

```
import com.google.inject.AbstractModule;

public class Module extends AbstractModule {

    @Override
    protected void configure() {

        bind(FinancialHooks.class).to(CustomFinancialHooks.class);
    }
}
```

### ***Third-party deduction hooks***

Use the available hooks in the interfaces

`curam.core.impl.PreCreateThirdPartyDeductionILIs` and

`curam.core.impl.PostCreateThirdPartyDeductionILIs` to perform custom business processing before and after the creation of third-party deduction instruction line items. For more information, see the Javadoc for the associated class.

### **Procedure**

1. Provide an implementation of the required interface. The following code shows a sample for the `PreCreateThirdPartyDeductionILIs` class.

```
import curam.core.struct.CaseDeductionItemDtls;
import curam.core.struct.CurrencyExchangeDtls;
import curam.core.struct.FCAmount;
import curam.core.struct.FinCompCoverPeriod;
import curam.core.struct.FinancialComponentDtls;
import curam.core.struct.TotalDeductionAmount;
import curam.util.exception.AppException;
import curam.util.exception.InformationalException;

public class CustomPreCreateThirdPartyDeductionILIs
    implements PreCreateThirdPartyDeductionILIs {

    @Override
    public void preCreate(final CaseDeductionItemDtls caseDeductionItemDtls,
        final FinancialComponentDtls fcDtls,
        final FinCompCoverPeriod fcCoverPeriod, final FCAmount fcAmount,
        final CurrencyExchangeDtls currencyExchangeDtls,
        final TotalDeductionAmount totalDeductionAmount)
        throws AppException, InformationalException {

        //Add custom code...
    }
}
```

2. Bind the implementation in a Guice module. The following code sample shows an example of how to bind the implementation:

```
import com.google.inject.AbstractModule;

public class Module extends AbstractModule {

    @Override protected void configure() {

        bind(PreCreateThirdPartyDeductionILIs.class).to(CustomPreCreateThirdPartyDeductionILIs.class);
    }
}
```

## Regenerate instruction line item hook

Use the available hook in

`curam.core.hook.impl.RegenerateInstructionLineItemHook` to prevent the regeneration of an instruction line item (ILI) when cancelled payments are reissued.

### About this task

By default, all ILIs that are included when a payment is generated are regenerated if the payment is reissued after cancellation. You can specify that a specific ILI such as an ILI for a third party deduction is not regenerated by providing an implementation of the `curam.core.hook.impl.RegenerateInstructionLineItemHook` customization point. For more information, see the Javadoc for the associated class.

### Procedure

1. Provide an implementation of the `RegenerateInstructionLineItemHook` class. The following code sample shows how to implement the `regenerateInstructionLineItemHook` class.

```
import curam.core.hook.impl.RegenerateInstructionLineItemHook;
import curam.core.struct.InstructionLineItemDtls;
import curam.util.exception.AppException;
import curam.util.exception.InformationalException;

public class CustomRegenerateInstructionLineItemHookTest
    implements RegenerateInstructionLineItemHook {

    @Override
    public boolean preRegenerateILI(final InstructionLineItemDtls iliDtls)
        throws AppException, InformationalException {

        // Add custom code...
    }
}
```

2. Bind the implementation in a Guice module. The following code sample shows how to bind the implementation.

```
import com.google.inject.AbstractModule;
import com.google.inject.multibindings.Multibinder;

public class Module extends AbstractModule {

    @Override
    protected void configure() {
        final Multibinder<InstructionLineItemEvents> eventListeners =
            Multibinder.newSetBinder(binder(), InstructionLineItemEvents.class);

        eventListeners.addBinding()
            .to(CustomInstructionLineItemPostInsertEvents.class);
    }
}
```

**Instruction line item post insert hook**

Use the available post insert hook on the `curam.core.sl.event.impl.InstructionLineItemEvents` abstract class to implement custom business processing such as updating custom entities.

**About this task**

For more information, see the Javadoc for the associated class and *Events* in the *Persistence Cookbook*.

**Procedure**

1. Provide an implementation of the `InstructionLineItemEvents` class. The following code sample shows how to implement the `InstructionLineItemEvents` class:

```
import curam.core.sl.event.impl.InstructionLineItemEvents;
import curam.core.struct.InstructionLineItemDtls;
import curam.util.exception.AppException;
import curam.util.exception.InformationalException;

public class CustomInstructionLineItemPostInsertEvents
    extends InstructionLineItemEvents {

    @Override
    public void
        postInsert(final InstructionLineItemDtls instructionLineItemDtls)
            throws AppException, InformationalException {

        // Add custom code...
    }
}
```

2. Bind the implementation in a Guice Module. The following code sample shows how to bind the implementation:

```
import com.google.inject.AbstractModule;
import com.google.inject.multibindings.Multibinder;

public class Module extends AbstractModule {

    @Override
    protected void configure() {
        final Multibinder<InstructionLineItemEvents> eventListeners =
            Multibinder.newSetBinder(binder(), InstructionLineItemEvents.class);

        eventListeners.addBinding()
            .to(CustomInstructionLineItemPostInsertEvents.class);
    }
}
```

**Payment instrument post insert and post modify hooks**

Use the available post insert and post modify hooks on the `curam.core.sl.event.impl.PaymentInstrumentInsertModifyEvents` abstract class to implement custom business processing such as updating custom entities.

**About this task**

For more information, see the Javadoc for the associated class and *Events* in the *Persistence Cookbook*.

## Procedure

1. Provide an implementation of the `PaymentInstrumentInsertModifyEvents` class. The following code sample shows how to implement the `PaymentInstrumentInsertModifyEvents` class:

```
import curam.core.sl.event.impl.PaymentInstrumentInsertModifyEvents;
import curam.core.struct.EffectiveDateReconcilStatusVersionNo;
import curam.core.struct.PIAmountEffectDate;
import curam.core.struct.PIInvalidatedVersionNo;
import curam.core.struct.PIREconcilStatusCode;
import curam.core.struct.PIREconcilStatusCodeInvalidatedVersionNo;
import curam.core.struct.PIStatusAmountEffectDate;
import curam.core.struct.PaymentInstrumentDtls;
import curam.core.struct.PaymentInstrumentKey;
import curam.util.exception.AppException;
import curam.util.exception.InformationalException;

public class CustomPaymentInstrumentInsertModifyEvents
    extends PaymentInstrumentInsertModifyEvents {

    @Override
    public void postInsert(final PaymentInstrumentDtls details)
        throws AppException, InformationalException {

        // Add custom code...
    }

    @Override
    public void postmodifyAmountEffectDate(
        final PaymentInstrumentKey pmtInstrumentKey,
        final PIAmountEffectDate piAmountEffectDate)
        throws AppException, InformationalException {

        // Add custom code...
    }

    @Override
    public void postmodifyEffectiveDateReconcilStatus(
        final PaymentInstrumentKey key,
        final EffectiveDateReconcilStatusVersionNo dtls)
        throws AppException, InformationalException {

        // Add custom code...
    }

    @Override
    public void postmodifyInvalidatedInd(final PaymentInstrumentKey key,
        final PIInvalidatedVersionNo details)
        throws AppException, InformationalException {

        // Add custom code...
    }

    @Override
    public void postmodifyReconcilStatusCode(
        final PaymentInstrumentKey paymentInstrumentKey,
        final PIREconcilStatusCode piStatusCodeVersionNo)
        throws AppException, InformationalException {

        // Add custom code...
    }

    @Override
    public void postmodifyReconcilStatusCodeInvalidatedInd(
        final PaymentInstrumentKey key,
        final PIREconcilStatusCodeInvalidatedVersionNo details)
        throws AppException, InformationalException {

        // Add custom code...
    }

    @Override
    public void postmodifyStatusAmountEffectDate(
        final PaymentInstrumentKey paymentInstrumentKey,
        final PIStatusAmountEffectDate piStatusAmountEffectDate)
        throws AppException, InformationalException {

        // Add custom code...
    }
}
}
```

2. Bind the implementation in a Guice module. The following code sample shows how to bind the implementation:

```
import com.google.inject.AbstractModule;
import com.google.inject.multibindings.Multibinder;

public class Module extends AbstractModule {

    @Override
    protected void configure() {
        final Multibinder<InstructionLineItemEvents> eventListeners =
            Multibinder.newSetBinder(binder(),
                PaymentInstrumentInsertModifyEvents.class);

        eventListeners.addBinding()
            .to(CustomPaymentInstrumentInsertModifyEvents.class);
    }
}
```

### 8.1.2.0 **Dynamic Offset hook**

The Dynamic Offset lives in the `curam.core.impl.DynamicOffset` interface. This hook point allows customers to determine an offset to the next processing date on financial components based on criteria such as a client's social security number. This offset can either be positive or negative. An example of its use could be facilitating the spreading of payments across a period if many citizens receive a particular benefit. The offset could also be static if the business requirements lend themselves to such an offset.

#### About this task

For more information, see the Javadoc for the associated class and *Events* in the *Persistence Cookbook*.

#### Procedure

1. Provide an implementation of the `DynamicOffset` interface. The following code sample shows how to implement the `DynamicOffset` interface:

```
import curam.core.impl.DynamicOffset;
import curam.core.struct.FCProcessingDtls;
import curam.util.exception.AppException;
import curam.util.exception.InformationalException;

public class CustomDynamicOffsetImpl extends DynamicOffset {

    /**
     * {@inheritDoc}.
     */
    @Override
    public short determineOffsetValue(final FCProcessingDtls details)
        throws AppException, InformationalException {

        // Add custom code...
    }
}
```

2. Bind the implementation in a Guice module. The following code sample shows how to bind the implementation:

```
import curam.core.impl.DynamicOffset;
import com.google.inject.AbstractModule;
import com.google.inject.multibindings.Multibinder;

public class Module extends AbstractModule {

    @Override
    protected void configure() {

        bind(DynamicOffset.class).to(CustomDynamicOffsetImpl.class);
    }
}
```

## Assessment customization points

You can extend financial functions that relate to assessments by using the customization points for assessments in Cúram.

### **Assessment engine hooks**

Extra financial hooks that relate to assessments are available by using the `curam.core.sl.infrastructure.assessment.impl.AssessmentEngineHooks` and `curam.core.sl.infrastructure.assessment.impl.ReassessEligibilityHook` classes.

The most important of these are `getRate`, `haltPaymentProcessing`, `manipulateFinancialComponents`, `rollUpComponents`, `getFCDatesAndValues`, `reassessEligibility`, `stopFinancialProcessing`, `filterNewCaseDecisions`, and `listCaseDecisionsForRegenerateFinancials`.

- **getRate**

The `getRate` method retrieves a rate that is used in the calculation of the deduction amount. The rate, if populated, overrides the rate that is specified on the financial component.

- **haltPaymentProcessing**

The `haltPaymentProcessing` method determines whether to generate an Instruction Line Item (ILI) from a financial component. The `haltPaymentProcessing` method provides a hook for customers so that they can halt generating payments on a case. The method returns `False` by default, indicating to continue payment processing as normal. Customized code can set the return to `True`, in which case the payment is halted. Ensure that the method considers only primary financial components and does not deduce financial components.

- **manipulateFinancialComponents**

The `manipulateFinancialComponents` method can change the list of financial components before it generates the financials. The default implementation returns the list unchanged.

- **rollUpComponents**

The `rollUpComponents` method rolls up matching contiguous components to minimize the number of identical, contiguous financial components on the system. If you have extra roll-up criteria that you want considered, you can specify it here.

- **getFCDatesAndValues**

A method hook is added to the method

`curam.core.impl.FCGenerationUtils.getFCDatesAndValues`, which can be used to override the value returned by the default implementation. You can use the method to apply rounding logic to payments amounts generated in the financial component.

You can use the hook by using Guice to bind an implementation to interface

`curam.core.sl.infrastructure.assessment.impl.FCGenerationHook`. The hook is optional. Only one implementation can be provided. There is no implementation for the hook at the Platform layer.

Your implementation of the method hook must return a valid value and not return null. If your implementation does not need to modify the default return value, then it must return the default value that was passed in through the first parameter `defaultResult`.

- **reassessEligibility**

The

`curam.core.sl.infrastructure.assessment.impl.ReassessEligibilityHook.reassessEligibility` hook that is provided allows implementers to perform custom processing at the point of assessing eligibility and also to determine whether to allow processing to continue as normal, or halt it when control returns to the `AssessmentEngine` code.

The `ReassessEligibilityHook` class supports custom implementations of hook points at various stages during the assessment processing, on a per product basis. Ensure that the custom implementation is configured by using the Guice dependency injection mechanism.

- **stopFinancialProcessing**

The

`curam.core.sl.infrastructure.assessment.impl.ReassessEligibilityHook.stopFinancialProcessing` hook that is provided here allows implementers to perform custom processing at the point of assessing eligibility to determine whether to halt the processing after it stores determinations and decisions and before it performs any financial calculations.

The `ReassessEligibilityHook` class supports custom implementations of hook points at various stages during the assessment processing, on a per product basis. Ensure that the custom implementation is configured by using the Guice dependency injection mechanism.

- **filterNewCaseDecisions**

The

`curam.core.sl.infrastructure.assessment.impl.ReassessEligibilityHook.filterNewCaseDecisions` hook that is provided here allows implementers to perform custom processing at the point of assessing eligibility to determine whether to filter any of the new decisions that are created by the current reassessment before the decisions are used in the payment or decision reconciliation process. The process is used to generate new financial components and overpayments, underpayments, or both.

The `ReassessEligibilityHook` class supports custom implementations of hook points at various stages during the assessment processing, on a per product basis. Ensure that the custom implementation is configured by using the Guice dependency injection mechanism.

- **listCaseDecisionsForRegenerateFinancials**

The

`curam.core.sl.infrastructure.assessment.impl.ReassessEligibilityHook.listCaseDecisionsForRegenerateFinancials`

hook point that is provided here allows the implementer to override the default list of decisions before the regenerate financial components processing uses the decisions.

The `ReassessEligibilityHook` class supports custom implementations of hook points at various stages during the assessment processing, on a per product basis. Ensure that the custom implementation is configured by using the Guice dependency injection mechanism.

### **Assessment engine events**

Other customization points are available in the form of events by using the `curam.core.sl.infrastructure.assessment.implAssessmentEngineEvent` class.

The most important of these customization points are `preReassessEligibility`, `postNormalReassessment`, `postReassessEligibility`, `postReassessFCs`, `reviewDecisionAndComponents`, `postInsertExamineDecisions`, `postProcessBenefitOverpayment`, `postProcessBenefitUnderpayment`, `postProcessLiabilityOverpayment`, `postProcessLiabilityUnderpayment`, and `postProcessPaymentCorrection`.

Listeners that implement the `AssessmentEngineEvent` interface can override the methods in the `curam.core.sl.infrastructure.assessment.implAssessmentEngineEvent` class to include extra custom processing.

- **`preReassessEligibility`**  
The `preReassessEligibility` method is called before the creation of financial components on initial case activation and also on the regeneration of financial components on case reassessment. The `preReassessEligibility` method is used for both classic and Cúram Express Rules (CER) enabled products.
- **`postNormalReassessment`**  
The `postNormalReassessment` method is called after the creation of financial components on initial case activation and also on the regeneration of financial components on case reassessment. The `postNormalReassessment` is used for classic products only and is not called for CER enabled products.
- **`postReassessEligibility`**  
The `postReassessEligibility` method is called after the creation of financial components on initial case activation and also on the regeneration of financial components on case reassessment. The `postReassessEligibility` method is used for both classic and CER enabled products.
- **`postReassessFCs`**  
The `postReassessFCs` method is called for both classic and CER enabled products when a reassessment takes place, that is, the case is reassessed because of a change, but an overpayment or underpayment might not necessarily be created.
- **`reviewDecisionAndComponents`**  
The `reviewDecisionAndComponents` method is called one time per decision, as the decision is being stored but before the value for the new `caseDecisionID` is set. The details of the new decisions can be inspected here before the details are committed.
- **`postInsertExamineDecisions`**  
The `postInsertExamineDecisions` method is called after newly created decision details are stored. Custom code that wants to store or link old and new decisions can do so by

overriding the hook and adding an alternative implementation. The event is fired one time per decision.

- **postProcessBenefitOverpayment**  
The `postProcessBenefitOverpayment` method is called after the creation of a benefit overpayment case during case reassessment. Custom processing that wants to store the newly created case details can be added here.
- **postProcessBenefitUnderpayment**  
The `postProcessBenefitUnderpayment` method is called after the creation of a benefit underpayment case during case reassessment. Custom processing that wants to store the newly created financial component or newly created case details, depending on which was created, can be added here.
- **postProcessLiabilityOverpayment**  
The `postProcessLiabilityOverpayment` method is called after the creation of a liability overbilling case during case reassessment. Custom processing that wants to store the newly created case details can be added here.
- **postProcessLiabilityUnderpayment**  
The `postProcessLiabilityUnderpayment` method is called after the creation of a liability underbill during case reassessment. Custom processing that wants to store the newly created financial component or newly created case details, depending on which was created, can be added here.
- **postProcessPaymentCorrection**  
The `postProcessPaymentCorrection` method is called after the creation of a payment correction case during case reassessment. Custom processing that wants to store the newly created case details can be added here.

# Notices

---

Permissions for the use of these publications are granted subject to the following terms and conditions.

## **Applicability**

These terms and conditions are in addition to any terms of use for the Merative website.

## **Personal use**

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of Merative

## **Commercial use**

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of Merative.

## **Rights**

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

Merative reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by Merative, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

MERATIVE MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Merative or its licensors may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents.

Information concerning non-Merative products was obtained from the suppliers of those products, their published announcements or other publicly available sources. Merative has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-Merative products. Questions on the capabilities of non-Merative products should be addressed to the suppliers of those products.

Any references in this information to non-Merative websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those

websites are not part of the materials for this Merative product and use of those websites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

The licensed program described in this document and all licensed material available for it are provided by Merative under terms of the Merative Client Agreement.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to Merative, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. Merative, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. Merative shall not be liable for any damages arising out of your use of the sample programs.

## ***Privacy policy***

---

The Merative privacy policy is available at <https://www.merative.com/privacy>.

## ***Trademarks***

---

Merative™ and the Merative™ logo are trademarks of Merative US L.P. in the United States and other countries.

IBM®, the IBM® logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Adobe™, the Adobe™ logo, PostScript™, and the PostScript™ logo are either registered trademarks or trademarks of Adobe™ Systems Incorporated in the United States, and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft™, Windows™, and the Windows™ logo are trademarks of Microsoft™ Corporation in the United States, other countries, or both.

UNIX™ is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.