



Technical Design for

FSD003.03 - Irrigation Deposit Distribution Algorithm

Table of Contents

[Table of Contents 2](#_Toc193497133)

[1 Solution Elements 4](#_Toc193497134)

[1.1 Business Requirements 4](#_Toc193497135)

[1.2 Issues / Concerns 4](#_Toc193497136)

[1.3 Assumptions 4](#_Toc193497137)

[1.4 Scope 4](#_Toc193497138)

[1.5 Solution Components 5](#_Toc193497139)

[1.5.1 Configuration Tools 5](#_Toc193497140)

[1.5.2 Deliverable Configurations 16](#_Toc193497141)

[1.5.3 Client Configurations 18](#_Toc193497142)

[2 Test Conditions 20](#_Toc193497143)

[3 Sign-Off 25](#_Toc193497144)

[4 Appendix 25](#_Toc193497145)

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Author** | **Notes** |
| 03/10/2025 | 0.1 | Bianca Magsakay | Initial Version. |
| 03/20/2025 | 0.2 | Bianca Magsakay | Addressed 1st pass review points and made necessary changes based on recent updates on FSD. |
| 03/26/2025 | 0.3 | Bianca Magsakay | Updated based on the recent updates on FSD. |
| 03/28/2025 | 0.4 | John Mari Alcausin | Updated during development |
| 04/02/2025 | 0.5 | John Mari Alcausin | Updated after 1st pass review |

**Related Documents**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Document Name** |
| 01/29/2025 | 1.0 | IPC\_FSD.003 - Irrigation Deposits.docx |
|  |  |  |

# Solution Elements

## Business Requirements

Idaho Power Company (IPC) requires to provide a deposit if they meet certain criteria through Irrigation Service Accounts. These Deposits need to be created at the start of the year, paid by start of season to begin Irrigation service and must be released at the end of the irrigation season. There are two types of Irrigation deposits, Tier 1 and Tier 2.

* The system shall have the ability to review irrigation (rate type) accounts and issue deposits refunds/credits on the account on a user defined schedule.  Irrigation customers usage is seasonal in nature.  Currently we collect security deposits at the beginning of the growing (pumping) season and hold the deposit until the end of the growing (pumping) season.  (currently reviewed in September)
* The system shall have the ability to review irrigation (rate type) accounts and issue deposits refunds/credits on the account with each monthly bill and the remainder of the deposit on a user defined schedule.  Tier two irrigation customers usage is seasonal in nature.  Currently we collect security deposits at the beginning of the growing (pumping) season and apply a portion of the deposit (including any accumulated interest) equal to the monthly bill.  Following the September billing remaining deposit is refunded to the customer at the end of the growing (pumping) season.
* The product shall have the ability to determine a deposit amount based on jurisdictional specific business rules, including: Multiple calculations of security deposit based on horsepower based on security deposit type.  Requires pump horsepower must be stored in Service Point.

## Issues / Concerns

N/A

## Assumptions

1. Irrigation SA is linked to one Irrigation Deposit SA at a time.

## Scope

This technical design will cover the Irrigation Deposit Distribution against Billed Charges Algorithm

## Solution Components

### Configuration Tools

#### Plug-In Script

##### CMIrrDepBl – Irrigation Deposit Distribution against Billed Charges (New)

|  |  |
| --- | --- |
| **Plug-In Script** | CMIrrDepBl |
| **Description** | Irrigation Deposit Distribution against Billed Charges |
| **Detailed Description** | This plug-in script will be responsible for maintaining Tier 2 Deposits by calculating any interest earned monthly, applying interest earned against any monthly charges and applying deposit balance against any remaining current charges on a monthly basis. |
| **Script Type** | Plug-In Script |
| **Algorithm Entity** | SA Type – Bill Completion |
| **Script Engine Version** | Groovy |

###### Groovy Members – Revision History

* Add as needed.

###### Groovy Members – Library Imports

* Add necessary imports.

###### Groovy Members – Soft Parameter Annotation

|  |  |  |  |
| --- | --- | --- | --- |
| **Sequence** | **Required** | **Entity** | **Variable Name** |
| 10 | Y | Characteristic Type | spIdCharType |
| 20 | Y | Bill Factor | irrigAnnualDepIntBillFactor |
| 30 | Y | Adjustment Type | interestRefundAdjType |
| 40 | Y | Adjustment Type | transferAdjType |
| 50 | Y | String | irrigOverpaymentSaType |
| 60 | Y | String | irrigSaTypes |
| 70 | Y | Adjustment Type | syncAdjustmentType |

###### Groovy Members – Constants, Variables and Library Scripts

* Define global variables
  + String irrigOverpaymentSaTypeParmDesc;
  + String irrigSaTypesParmDesc;
  + String spIdCharTypeParmDesc;
  + ServiceAgreement\_Id *irrigationSaId*
  + ServiceAgreement\_Id *depositSaId*
* Define constants
  + private static final String COL\_SA\_TYPE\_CD = "SA\_TYPE\_CD";
  + private static final String COL\_SA\_ID = "SA\_ID";
* private static final String FLD\_CHAR\_LBL = "CHARACTERIST\_LBL";Define any library that is used
  + private CMSParValLib **parmValHelper** = createLibraryScript(CMSParValLib.class)
  + private CMIpcCommLib **commLib** = createLibraryScript(CMIpcCommLib.class)
* Update as needed

###### Groovy Members – Main Processing

* Set *depositSaId* to hard parameter serviceAgreement.getId()
* Set *irrigationSaId* to [*retrieveIrrigSaLinkedToDepositSa(ServiceAgreement\_Id saId,CharateristicType charType,String saTypes)*](#_retrieveIrrigSaLinkedToDepositSa(Se)
* If *depositSaId.getEntity().getDepositInterestCalculated()* matches the current date
  + Terminate process

//Main Process

* Initialize *depositInterestAmount* to zero
* If depositSaId*.getEntity().getDepositInterestCalculated()* is null
  + Set *startDate* to *depositSaId.getStartDate()*
* Else
  + Set *startDate* to *depositSaId.getEntity().getDepositInterestCalculated()*
* Set *depositInterestAmount* to [*getDepositSaInterest(depositSaId,startDate)*](#_getDepositSaInterest(ServiceAgreeme)
* If *depositInterestAmount* is NOT credit
  + Terminate Process
* If *depositInterestAmount.negate()* is less than *irrigationSaId.getEntity().getCurrentBalance()*
  + *commLib.createAdjustment(irrigationSaId,null,interestRefundAdjType,depositInterestAmount)*
  + Set *remainingBalance* to *irrigSaCurrBalance* minus *depositInterestAmount.negate()*
  + Set *irrigOverpaymentSaId* to [*retrieveIrrigOverpaymentSa(irrigationSaId.getAccount(),irrigOverpaymentSaType)*](#_retrieveIrrigOverpaymentSa(Account__1)
  + If Irrigation Overpayment SAexists
    - Set *overpaymentSaCurrBal* to *irrigOverpaymentSaId.getEntity().getCurrentBalance().negate()*
    - If *overpaymentSaCurrBal* is greater than zero
      * If *overpaymentSaCurrBal* is greater than *remainingBalance*
        + *commLib.createAdjustment (irrigOverpaymentSaId,irrigationSaId,transferAdjType,remainingBalance.negate())*
        + Set *remainingBalance* to zero
      * Else
        + *overpaymentSaCurrBal = overpaymentSaCurrBal.negate()*
        + *commLib.createAdjustment (irrigOverpaymentSaId,irrigationSaId,transferAdjType,overpaymentSaCurrBal.negate())*
        + Set *remainingBalance* minus *overpaymentSaCurrBal*
  + If *remainingBalance* is greater than zero
    - *Set depositSaCurrBal* to *depositSaId.getEntity().getPayoffBalance().negate()*
      * If *depositSaCurrBal* is greater than zero and is greater than *remainingBalance*
        + *commLib.createAdjustment (depositSaId,irrigationSaId,transferAdjType,remainingBalance.negate())*
        + *commLib.createAdjustment (depositSaId,null,syncAdjustmentType,remainingBalance.negate())*
        + Update depositSaId’s Total Amount To Bill to (*depositSaCurrBal.negate minus remainingBalance)*

|  |
| --- |
| Sample Code:  newTotalAmtToBill = depositSaCurrBal.negate().subtract(remainingBalance)  saDto.setTotalAmountToBill(newTotalAmtToBill) |

* + - * Else if *depositSaCurrBal*.negate() is greater than zero andis less than *remainingBalance*
        + *commLib.createAdjustment (depositSaId,irrigationSaId,transferAdjType,depositSaCurrBal.negate())*
        + *commLib.createAdjustment (depositSaId,null,syncAdjustmentType,depositSaCurrBal.negate())*
* If *depositInterestAmount.negate()* is greater than *irrigSaCurrBalance*
  + If irrigSaCurrBalance is positive
    - *commLib.createAdjustment (irrigationSaId,null,interestRefundAdjType,irrigSaCurrBalance.negate())*
  + Set *remainingInterest* to *depositInterestAmount.negate()* minus *irrigSaCurrBalance*
  + Set *irrigOverpaymentSaId* to [*retrieveIrrigOverpaymentSa(irrigationSaId.getAccount(),irrigOverpaymentSaType)*](#_retrieveIrrigOverpaymentSa(Account__1)
  + If Irrigation Overpayment SAexists
    - *commLib.createAdjustment (irrigOverpaymentSaId,null,interestRefundAdjType,remainingInterest.negate())*
  + Else
    - Create an Overpayment SA using ServiceAgreementCreator class
      * Account Id = *irrigationSaId*’s Account Id
      * SA Type = *irrigOverpaymentSaType*
      * CIS Division = *irrigationSaId*’s CIS Division
      * Start Date = Process Date
      * Customer Read = false
    - *commLib.createAdjustment (irrigOverpaymentSaId,null,interestRefundAdjType,remainingInterest.negate())*
* Update *depositSaId*’s Interest Calculation date to Process Date

*// End of Process*

###### Groovy Members – Methods

retrieveIrrigSaLinkedToDepositSa(ServiceAgreement\_Id saId, CharateristicType charType, String saTypes)

|  |
| --- |
| SQL:  SELECT SA.SA\_ID  FROM CI\_SA\_SP SASP, CI\_SA SA  WHERE TRIM(SA.SA\_TYPE\_CD) IN (:saTypeList)  AND SA.SA\_STATUS\_FLG = ‘20’  AND SA.CIS\_DIVISION = :cisDiv  AND SA.SA\_ID = SASP.SA\_ID  AND SASP.SP\_ID IN (SELECT SAC.CHAR\_VAL\_FK1      FROM CI\_SA\_CHAR SAC      WHERE SAC.SA\_ID = :saId      AND SAC.CHAR\_TYPE\_CD = :charTypeCd)  AND SASP.START\_DTTM <= :processDate  AND (SASP.STOP\_DTTM IS NULL OR SASP.STOP\_DTTM > :processDate) |

* Set :charTypeCd to charType
* Set :saId FK1 to String *saId*
* Set :cisDiv to *saId’s* CIS Division
* Set saTypeList to *saTypes* (Use *commLib.formatListOfValues()* to make this string in valid format)
* Set :processDate to Process Date
* Return SA\_ID

getDepositSaInterest(ServiceAgreement sa, Date beginDate)

* Set *interestRateVal* to [*retBillFactorValue()*](#_getSaCurrentBalance(ServiceAgreemen)
* Initialize variables
  + Set *interestAmount* to zero
  + Set *totalInterestAmount* to zero
  + Set *previousArrearsDate* to null
  + Set *payOffAmount* to zero
  + Set *principle* to zero
* Loop through [*getFts(beginDate,sa.getId())*](#_getFts(Date_beginDate,_ServiceAgree)– use SQLResultRow row
  + If *previousArrearsDate* is not null AND *previousArrearsDate* is after *beginDate*
    - Set *ftStartDate* to *previousArrearsDate*
  + Else
    - Set *ftStartDate* to *beginDate*
  + Set *payoffAmount* to sa.getFinancialBalances(*ftStartDate*, *ftStartDate*, Bool.TRUE).getTotalAmount()
  + If *payoffAmount* is positive
    - Set *payoffAmount* to zero
  + Set *endDate* to row.getDate("ARS\_DT")
  + Set *principle* to *payoffAmount*
  + Set *interestAmount* to [*getInterestAmount(ftStartDate,endDate,principle,interestRateVal)*](#_getInterestAmount(Date_startDate,_D)
  + Set *previousArrearsDate* to *endDate*
  + Set *totalInterestAmount* to sum of *totalInterestAmount* and *interestAmount*
* If *previousArrearsDate* is null OR *previousArrearsDate* is not equal to Process Date
  + If *previousArrearsDate* is null OR *beginDate* is after *previousArrearsDate*
    - Set *ftStartDate* to *beginDate*
  + Else
    - Set *ftStartDate* to *previousArrearsDate*
  + Set *payoffAmount* to sa.getFinancialBalances(*ftStartDate*, *ftStartDate*, Bool.TRUE).getTotalAmount()
  + If *payoffAmount* is positive
    - Set *payoffAmount* to zero
  + Set *endDate* to Process Date
  + Set *principle* to *payoffAmount*
  + Set *interestAmount* to [*getInterestAmount(ftStartDate,endDate,principle,interestRateVal)*](#_getInterestAmount(Date_startDate,_D)
  + Set *totalInterestAmount* to sum of *totalInterestAmount* and *interestAmount*

|  |
| --- |
| Sample Code:  List<SQLResultRow> ftList = getFt(beginDate);  for(SQLResultRow row: ftList){    // Start Date is the greater of the Begin Date or the Previous Arrears Date  if(notNull(previousArrearsDate) && previousArrearsDate.isAfter(beginDate)){  ftStartDate = previousArrearsDate;  }else{  ftStartDate = beginDate;  }  //Get Current Balance  payoffAmount = sa.getFinancialBalances(ftStartDate, ftStartDate, Bool.TRUE).getTotalAmount()  if(payoffAmount.isPositive()){  payoffAmount = Money.ZERO;  }  //Set Arrears Date as End Date  endDate = row.getDate("ARS\_DT");  principle = payoffAmount;  //Calculate Interest Amount  interestAmount = getInterestAmount(ftStartDate, endDate, principle, interestRate);  previousArrearsDate = endDate;  totalInterestAmount = totalInterestAmount.add(interestAmount);  }  // If the last processed arrears date is different than interest calculation period end date, calculate interest as of the calculation period end date.    if((isNull(previousArrearsDate) || !processDate.equals(previousArrearsDate))){    if(isNull(previousArrearsDate) || beginDate.isAfter(previousArrearsDate)){  ftStartDate = beginDate;  }else{  ftStartDate = previousArrearsDate;  }  //Get Current Balance  payoffAmount = sa.getFinancialBalances(ftStartDate, ftStartDate, Bool.TRUE).getTotalAmount()  if(payoffAmount.isPositive()){  payoffAmount = Money.ZERO;  }  //Set Arrears Date as End Date  endDate = processDate;  principle = payoffAmount;  //Calculate Interest Amount  interestAmount = getInterestAmount(ftStartDate, endDate, principle, interestRate);    totalInterestAmount = totalInterestAmount.add(interestAmount);  } |

getBillFactorValue()

* Initialize *interestRate* to zero
* Create new instance for BillFactorValueInfoRetriever
* Create new instance for BillFactorValueInfoRetrieverData
* Create new instance for ApplyRateData
* Set ApplyRateData’s accounting date to Process Date
* Set the following values for BillFactorValueInfoRetrieverData
  + To Get Bill Factor = True
  + Bill Factor = irrigAnnualDepIntBillFactor soft parameter value
  + Bill Factor Period Start Date = Process Date
  + Bill Factor Period End Date = Process Date
  + No Proration Accounting Date = Process Date
  + No Proration Start Date = Process Date
  + No Proration End Date = Process Date
  + To Fill the Gap = False
  + Service Agreement = Deposit SA
  + If Characteristic Premise on Deposit SA is not null
    - Char Source Premise = Characteristic Premise on Deposit SA
  + Apply Rate Data = ApplyRateData
* Call retrieveBillFactorValueInfo from BillFactorValueInfoRetriever instance passing BillFactorValueInfoRetrieverData
* Get Bill Factor Value Data List from BillFactorValueInfoRetrieverData
* If Bill Factor Value Data List is not null
  + Set *interestRate* to the first entry on the list

|  |
| --- |
| Sample Code:  BigDecimal interestRate = BigDecimal.ZERO;  BillFactorValueInfoRetriever bfValueinfoRetriever = BillFactorValueInfoRetriever.Factory.newInstance();  BillFactorValueInfoRetrieverData bfValueInfo = BillFactorValueInfoRetrieverData.Factory.newInstance();  ApplyRateData applyRateData = ApplyRateData.Factory.newInstance();  applyRateData.setAccountingDate(getProcessDateTime().getDate());  bfValueInfo.setToGetBillFactor(Bool.TRUE);  bfValueInfo.setBillFactor(irrigAnnualDepIntBillFactor);  bfValueInfo.setBillFactorPeriodStartDate(processDate);  bfValueInfo.setBillFactorPeriodEndDate(processDate);  bfValueInfo.setNoProrationAccountingDate(processDate);  bfValueInfo.setNoProrationStartDate(processDate);  bfValueInfo.setNoProrationEndDate(processDate);  bfValueInfo.setToFillTheGap(Bool.FALSE);  bfValueInfo.setServiceAgreement(depositSaId.getEntity());  if(notNull(charPremise)){  bfValueInfo.setCharSourcePremise(charPremise);  }  bfValueInfo.setApplyRateData(applyRateData);  bfValueinfoRetriever.retrieveBillFactorValueInfo(bfValueInfo);  List<BillFactorValueData> bfValueInfoCollection = bfValueInfo.getBillFactorValueInfoData().getBillFactorValueDataList();  if(!bfValueInfoCollection.isEmpty()){  interestRate = bfValueInfoCollection.get(0).getBfValue();  } |

retrieveIrrigOverpaymentSa(Account\_Id acctId, ServiceAgreementType saType)

|  |
| --- |
| SQL:  SELECT SA.SA\_ID  FROM CI\_SA SA  WHERE SA.ACCT\_ID = :acctId  AND SA.SA\_TYPE\_CD = :saTypeCd  AND SA.SA\_STATUS\_FLG = ‘20’ |

* Set :acctId to *acctId*
* Set :saTypeCd to *saType*
* If result is not null
  + Return SA\_ID
* Else
  + Return null

getFts(Date beginDate, ServiceAgreement\_Id saId)

|  |
| --- |
| SQL to retrieve SA FT’s  SELECT DISTINCT(FT.ARS\_DT)  FROM CI\_FT FT  WHERE FT.SA\_ID = :saId  AND FT.FT\_TYPE\_FLG IN ('AD','PS')  AND FT.ARS\_DT BETWEEN :startDate AND :currentDate  AND FT.FREEZE\_SW = ‘Y’  ORDER BY FT.ARS\_DT |

* Set :saId to *saId*
* Set :startDate to *beginDate*
* Set :currentDate to Current Date
* Return list result

getInterestAmount(Date startDate, Date endDate, Money principle, BigDecimal interestRate)

* Create instance for InterestCalculator
* Set *interestAmount* to InterestCalculator’s calculateDepositInterest() passing the following parameters
  + startDate
  + endDate
  + principle
  + interestRate

Sample code:

|  |
| --- |
| Money interestAmount;  InterestCalculator calculator = InterestCalculator.Factory.newInstance();  interestAmount = calculator.calculateDepositInterest(startDate, endDate, principle, interestRate);  return interestAmount; |

createAdjustment(ServiceAgreement\_Id saId, ServiceAgreement\_Id transferSaId, AdjustmentType adjustmentType, Money adjustmentAmt)

* Create Adjustment with the following parameters
  + SA Id = *saId*
  + Adjustment Type = *adjustmentType*
  + If *transferSaId* is not null
    - Transfer SA Id = *transferSaId*
  + Adjustment Amount = *adjustmentAmt*

###### Groovy Members – Extra Soft Parameter Validation

* Validate irrigOverpaymentSaType using SQL below
* Validate each value from *irrigSaTypes*, delimited by constant COMMA using SQL below. Use split.

|  |
| --- |
| SELECT SA\_TYPE\_CD FROM CI\_SA\_TYPE  WHERE SA\_TYPE\_CD = :saTypeCd |

* + If not a valid SA Type, raise [MSG1](#_CMCONSRV-PR_(New))
* Validate spIdCharType by calling validateAlgParmCharTypeEntity() from **parmValHelper** passing the following:
  + spIdCharType parameter description
  + spIdCharType parameter value
  + CharacteristicEntityLookup.constants.SA

### Deliverable Configurations

#### Algorithm Type

##### CM-IRRDEPBL (New)

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm Type** | | CM-IRRDEPBL | |
| **Description** | | Irrigation Deposit Distribution against Billed Charges | |
| **Detailed Description** | | This algoirithm will be responsible for maintaining Tier 2 Deposits by calculating any interest earned monthly, applying interest earned against any monthly charges and applying deposit balance against any remaining current charges on a monthly basis. | |
| **Algorithm Entity** | | SA Type – Bill Completion | |
| **Program Type** | | Plug-in Script | |
| **Plug-in Script** | | [CMIrrDepBl](#_CMIrrDepBl_–_Irrigation) | |
| **Parameters** | | | |
| **Sequence** | **Parameter** | | **Required** |
| 10 | SP ID Characteristic Type | | Y |
| 20 | Irrigation Annual Deposit Interest Bill Factor | | Y |
| 30 | Interest Refund Applied Adjustment Type | | Y |
| 40 | Transfer Adjustment Type | | Y |
| 50 | Irrigation Overpayment SA Type | | Y |
| 60 | Irrigation SA Types (comma-delimited) | | Y |
| 70 | Sync Adjustment Type | | Y |

#### Message

Technical Note: Use the existing message if it already exists.

|  |  |  |  |
| --- | --- | --- | --- |
| **Message Category** | | TBD | |
| **Description** | | Implementer’s Message | |
| **Message Number** | **Message Text** | | **Detailed Description** |
| MSG1 | %1 algorithm parameter (%2) is not a valid SA Type | | %1 – Parameter Description  %2 – SA Type Value |

#### Field

##### CM\_OVP\_SA\_LBL

|  |  |
| --- | --- |
| **Field Name** | CM\_OVP\_SA\_LBL |
| **Data Type** | Character |
| **Field Precision** | 1 |
| **Description** | Overpayment SA Type |

##### CM\_IRR\_SA\_LBL

|  |  |
| --- | --- |
| **Field Name** | CM\_IRR\_SA\_LBL |
| **Data Type** | Character |
| **Field Precision** | 1 |
| **Description** | Irrigation SA Type |

### Client Configurations

#### Adjustment Type

##### IRRINTAP (New)

|  |  |
| --- | --- |
| **Field Name** | **Value** |
| Adjustment Type | IRRINTAP |
| Description | Interest Refund Applied |
| CIS Division |  |
| Adjustment Amount Type | Non-Calculated Amount |
| Distribution Code | TBD |
| Print By Default | Yes |
| Adjustment Freeze Option | Freeze At Will |

##### IRRDXFER (New)

|  |  |
| --- | --- |
| **Field Name** | **Value** |
| Adjustment Type | IRRDXFER |
| Description | Irrigation Deposit Transfer |
| CIS Division |  |
| Adjustment Amount Type | Non-Calculated Amount |
| Distribution Code | TBD |
| Print By Default | Yes |
| Adjustment Freeze Option | Freeze At Will |

#### Algorithm

##### CM-IRRDEPBL (New)

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm Code** | CM-IRRDEPBL | | |
| **Description** | Irrigation Deposit Distribution against Billed Charges | | |
| **Algorithm Type** | [CM-IRRDEPBL](#_CM-IRRDEPBL_(New)) | | |
| **Parameters** | | | |
| **Parameter** | | **Sequence** | **Value** |
| SP ID Characteristic Type | | 10 | C1-SP |
| Annual Deposit Interest Bill Factor | | 20 | [IRRDEP](#_Irrigation_Deposit_Interest) |
| Interest Refund Applied Adjustment Type | | 30 | [IRRINTAPLY](#_IRRINTAPLY_(New)) |
| Transfer Adjustment Type | | 40 | [IRRDEPXFER](#_IRRDEPXFER_(New)) |
| Irrigation Overpayment SA Type | | 50 | [IRRXSCRDT](#_IRRXSCRDT_(New)) |
| Irrigation SA Types (comma-delimited) | | 60 | IEIRR,OEIRR |
| Sync Adjustment Type | | 70 | SYNC |

#### Bill Factor

##### Irrigation Deposit Interest Rate

|  |  |
| --- | --- |
| **Bill Factor** | IRRDEP |
| **Description** | Irrigation Deposit Interest Rate |
| **Bill Factor Type** | Regular |
| **Value Type** | Percentage |
| **BF Characteristic Values** | |
| **Effective Date** | **Value** |
| TBD | TBD |

#### SA Type

##### IRRXSCRD (New)

|  |  |
| --- | --- |
| **SA Type** | IRRXSCRD |
| **Description** | Irrigation Excess credit (overpayment) |
| **Deposit Class** | Irrigation |

##### IIRRDPT2 (New)

|  |  |  |  |
| --- | --- | --- | --- |
| **SA Type** | | IIRRDPT2 | |
| **CIS Division** | | Idaho | |
| **Description** | | Idaho Irrigation Deposit – Tier 2 | |
| **Deposit Class** | | [Irrigation](#_Irrigation_(New)) | |
| **Characteristics** |  | |  |
| **Characteristic Type** | **Seq** | | **Value** |
| [IRRDEPTYP](#_IRRDEPTYP_(New)) | 10 | | TIER2 |
| **System Event** | **Seq** | | **Value** |
| Bill Completion | 10 | | [CM-IRRDEPBL](#_CM-IRRDEPBL) |

##### OIRRDPT2 (New)

|  |  |  |  |
| --- | --- | --- | --- |
| **SA Type** | | OIRRDPT2 | |
| **CIS Division** | | Oregon | |
| **Description** | | Oregon Irrigation Deposit – Tier 2 | |
| **Deposit Class** | | [Irrigation](#_Irrigation_(New)) | |
| **Characteristics** |  | |  |
| **Characteristic Type** | **Seq** | | **Value** |
| [IRRDEPTYP](#_IRRDEPTYP_(New)) | 10 | | TIER2 |
| **System Event** | **Seq** | | **Value** |
| Bill Completion | 10 | | [CM-IRRDEPBL](#_CM-IRRDEPBL) |

# Test Conditions

Below is a table outlining the functional scenarios and their expected results:

|  |  |  |
| --- | --- | --- |
| **#** | **Test Scenario** | **Expected Results** |
| 1 | Soft parameter validation:  Parameter Irrigation Overpayment SA Type is an invalid SA Type | Irrigation Overpayment SA Types parameter value is not a valid SA Type error message is displayed |
| 2 | Soft parameter validation:  Parameter Irrigation SA Types has invalid SA Type | Irrigation SA Types parameter value is not a valid SA Type error message is displayed |
| 3 | Soft parameter validation:  SP ID Characteristic Type parameter value is not a valid characteristic type for SA Entity | %1 algorithm parameter (%2) must be valid for the %3 entity error message is displayed |
| 4 | Customer with one Tier 2 deposit is Billed on a regular Monthly Bill.  Sample Scenario:  Irrigation SA: $100 Current Balance  Deposit SA’s Average Daily Balance = $-90  Bill factor value = 72%  Overpayment SA: $-20 Current Balance  Deposit SA: $-90 Payoff Balance | Interest Refund Applied Adjustment is applied to Irrigation SA   * Adjustment Amount = -5.33 * Remaining Balance = 94.67   Irrigation Deposit Transfer Adjustment is applied from Irrigation Excess credit SA to Irrigation SA   * Adjustment Amount = 20 * Remaining Balance = 74.67   Irrigation Deposit Transfer Adjustment is applied from Irrigation Deposit SA to Irrigation SA   * Adjustment Amount = 74.67   Irrigation SA’s Current Balance becomes zero  Irrigation Deposit SA’s payoff balance is -15.33  Irrigation Deposit SA’s Interest Calc Date is set to Current Date  Irrigation Excess credit SA’s Current Balance becomes 0 |
| 5 | Customer with one Tier 2 deposit is Billed on a regular Monthly Bill.  Sample Scenario:  Irrigation SA: $50 Current Balance  Deposit SA’s Average Daily Balance = $-60  Bill factor value = 72%  Overpayment SA: $-70 Current Balance  Deposit SA: $-60 Payoff Balance | Interest Refund Applied Adjustment is applied to Irrigation SA   * Adjustment Amount = -3.55 * Remaining Balance = 46.45   Irrigation Deposit Transfer Adjustment is applied from Irrigation Excess credit SA to Irrigation SA   * Adjustment Amount = 46.45 * Remaining Balance = 0   NO Irrigation Deposit Transfer Adjustment is applied from Irrigation Deposit SA to Irrigation SA  Irrigation SA’s Current Balance becomes zero  Irrigation Deposit SA’s payoff balance remains $-60  Irrigation Deposit SA’s Interest Calc Date is set to Current Date  Irrigation Excess credit SA’s Current Balance becomes $-23.55 |
| 6 | Customer with one Tier 2 deposit is Billed on a regular Monthly Bill.  Sample Scenario:  Irrigation SA: $20 Current Balance  Deposit SA’s Average Daily Balance = $-500  Bill factor value = 72%  Overpayment SA: $-10 Current Balance  Deposit SA: $-500 Payoff Balance | Interest Refund Applied Adjustment is applied to Irrigation SA   * Deposit Interest = -29.59 * Adjustment Amount = -20 just to cover Irrigation SA’s current balance * Remaining Interest = -9.59   Interest Refund Applied Adjustment is applied to Irrigation Excess credit SA   * Adjustment Amount = -9.59   Irrigation SA’s Current Balance becomes zero  Irrigation Excess credit SA’s current balance becomes $-19.59  Irrigation Deposit SA’s Interest Calc Date is set to Current Date |
| 7 | Customer with one Tier 2 deposit is Billed on a regular Monthly Bill.  Sample Scenario:  Irrigation SA: $200 Current Balance  Deposit SA’s Average Daily Balance = $-90  Bill factor value = 72%  Overpayment SA: $-20 Current Balance  Deposit SA: $-90 Payoff Balance | Interest Refund Applied Adjustment is applied to Irrigation SA   * Adjustment Amount = -5.33 * Remaining Balance = 194.67   Irrigation Deposit Transfer Adjustment is applied from Irrigation Excess credit SA to Irrigation SA   * Adjustment Amount = 20 * Remaining Balance = 174.67   Irrigation Deposit Transfer Adjustment is applied from Irrigation Deposit SA to Irrigation SA   * Adjustment Amount = 90   Irrigation SA’s remaining Current Balance is $84.67  Irrigation Deposit SA’s Interest Calc Date is set to Current Date |
| 8 | Customer with one Tier 2 deposit is Billed on a regular Monthly Bill.  Sample Scenario:  Irrigation SA: $20 Current Balance  Deposit SA’s Average Daily Balance = $-500  Bill factor value = 72%  Current Balance  Deposit SA: $-500 Payoff Balance | Interest Refund Applied Adjustment is applied to Irrigation SA   * Deposit Interest = -29.59 * Adjustment Amount = -20 just to cover Irrigation SA’s current balance * Remaining Interest = -9.59   Irrigation Excess credit SA is created  Irrigation Refund Applied Adjustment is applied to Irrigation Excess credit SA   * Adjustment Amount = -9.59   Irrigation SA’s Current Balance becomes zero  Irrigation Deposit SA’s Payoff Balance remains $-150  Irrigation Deposit SA’s Interest Calc Date is set to Current Date  Irrigation Excess credit SA’s Current Balance becomes $-9.59 |
| 9 | Customer with multiple Tier 2 deposits is Billed on regular Monthly Bill.  Sample Scenario:  Irrigation SA 1 / Irrigation Deposit SA 1  Irrigation SA: $100 Current Balance  Deposit SA’s Average Daily Balance = $-90  Deposit SA: $-90 Payoff Balance  Irrigation SA 2 / Irrigation Deposit SA 2  Irrigation SA: $10 Current Balance  Deposit SA’s Average Daily Balance = $-200  Deposit SA: $-200 Payoff Balance  Bill factor value = 72%  Overpayment SA: $-120 Current Balance | Irrigation SA 1 / Irrigation Deposit SA 1  Interest Refund Applied Adjustment is applied to Irrigation SA1   * Adjustment Amount = -5.33 * Remaining Balance = 94.67   Irrigation Deposit Transfer Adjustment is applied from Irrigation Excess credit SA to Irrigation SA1   * Adjustment Amount = 94.67   NO Irrigation Deposit Transfer Adjustment is applied from Irrigation Deposit SA1 to Irrigation SA1  Irrigation SA1’s Current Balance becomes zero  Irrigation Deposit SA1’s Payoff Balance is still $-90  Irrigation Excess credit SA becomes $-25.33  Irrigation Deposit SA1’s Interest Calc Date is set to Current Date  Irrigation SA 1 / Irrigation Deposit SA 1  Interest Refund Applied Adjustment is applied to Irrigation SA2   * Deposit Interest = -11.84 * Adjustment Amount = -10 just to cover Irrigation SA2’s current balance * Remaining Interest = -1.84   Interest Refund Applied Adjustment is applied to Irrigation Excess credit SA   * Adjustment Amount = -1.84   Irrigation SA2’s Current Balance becomes zero  Irrigation Deposit SA2’s Payoff Balance is still $-200  Irrigation Excess credit SA’s current balance becomes $-27.17  Irrigation Deposit SA2’s Interest Calc Date is set to Current Date |
| 10 | Customer with one Tier 2 deposit is Billed on a regular Monthly Bill. Deposit SA has interest calculation date prior to completion and there are multiple FTs within the next calculation period. | Interest is calculated from the previous calculation date up to the date of completion based on transactions between those dates. |
| 11 | Bill for Customer with Tier 2 deposits has been cancelled. Cancelled bill had Interest Refund Applied and Irrigation Deposit Transfer Adjustments. | Adjustments will still be associated with the cancelled bill. New adjustments will be created for the new bill. |
| 12 | Frozen bill segment for Customer with Tier 2 deposit is cancelled | The interest adjustment from the previous bill will be retained in the previous bill  *Note: A new bill segment will have to be manually created and picked up in the new bill* |

# Sign-Off

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Position** | **Date** | **Signature** |
| Rea Catibag | Principal Consultant | 03/24/2025 |  |
|  |  |  |  |

# Appendix