Aeos® Transactional Purge

Transactional (TRX) Purge consists of two processes that run periodically to remove data from the Aeos HealthcareWorkManagement (HCWM), BatchImport, and MessageDB databases as it becomes obsolete and unnecessary. It is needed to prevent capacity and performance issues that arise as the database grows over time. TRX Purge is an automated solution that is configured by modifying SQL Server Agent jobs or a table in the database.

About this Document

This Huron Healthcare internal document is for DevOPs/ PIT/TECS team members who need to configure or troubleshoot the TRX purge. It assumes that the reader has experience with configuring the Aeos product components and supporting Microsoft technologies.

Components

TRX Purge has two separate parts: Account Purge and Account Billing Cycle Purge (ABC Purge). Each part is a stored procedure controlled by a SQL Server Agent job.

Shared Tables

Both Account Purge and ABC Purge use the following tables for configuration settings, history, and error logs.

- Control.dbo.PurgeConfigSetting
- Control.dbo.PurgeHistory
- Control.dbo.PurgeCandidate
- Control.dbo.ExceptionLog

Account Purge

Account Purge consists of the following components:

- The SQL Server Agent Job: TransactionalPurge

  The TransactionalPurge SQL Server Agent job executes the account purge stored procedure. It is located on the APP DB server of the Aeos installation.

- Account Purge stored procedure: PurgeAccounts.

  The PurgeAccounts stored procedure in the HealthcareWorkManagement database.

- The shared purge tables listed above, as well as its own account-level detail for the batch level records in the PurgeHistory table: Control.dbo.PurgeAccountHistoryDetail.

  Account-level detail for the batch level records in the PurgeHistory table.

ABC Purge

ABC Purge consists of the following components:

- The SQL Server Agent Job: TransactionalABCPurge

  It is also located on the APP DB server of the Aeos installation.

- ABC Purge stored procedure: PurgeAccountBillingCycle

  The PurgeAccountBillingCycle stored procedure in the HealthcareWorkManagement database.

- The shared purge tables listed above, as well as its own account billing cycle-level detail for the batch level records in the PurgeHistory table: Control.dbo. PurgeABCHistoryDetail.

Starting Purge

By default, TRX Purge is installed but disabled. To turn it on, confirm the settings in the PurgeConfigSettings table and TransactionalPurge/TransactionalABCPurge SQL Server Agent job schedules, and then enable the jobs. Before you start TransactionalPurge, please read Initial Analysis/Set Up.

You can also run the purge manually by running the PurgeAccounts and PurgeAccountBillingCycle stored procedures in SSMS. The commands to run these stored procedures are EXEC HealthCareWorkManagement.dbo.PurgeAccounts and EXEC HealthCareWorkManagement.dbo.PurgeAccountBillingCycle respectively. They do not require any parameters.

Already Purging Check

When starting up, both these stored procedures check the PurgeHistory.TargetDatabase and PurgeHistory.RunStatus fields to make sure that no other purge is already running on the HCWM database. If another purge is running, the second purge process does not run past this check and ends with an error. An entry for this error is logged to Control.dbo.ExceptionLog.

Account Purge Process

A SQL Server Agent job called TransactionalPurge runs the PurgeAccounts sproc at a scheduled interval. By default, it runs weekly every Saturday at 9:00 am.

1. The PurgeAccounts sproc identifies purge-eligible records at the beginning of the run based on the following properties of the parent account. All the following must be true for an account to be purge-eligible (all numbers shown are default values of the settings in the PurgeConfigSetting table; setting names are shown as *green italics*):
    1. Account.AccountBalance has been zero for 180 consecutive days for accounts without denials (*AccountLastZeroedWithoutDenials*) or 365 consecutive days for accounts with denials (*AccountLastZeroedWithDenials*) based on Account.AccountBalanceZeroedDate.
    2. Account's action items without denials were last touched more than 180 days ago (*ActionLastTouchedWithDenials*), or action items with denials were last touched more than 365 days ago (*ActionLastTouchedWithDenials*).
    3. Account has no associated open action items - OR - account has NO action items but is purge-eligible.
    4. Account does not have a payer balance (BillingCycleSponsorPlan)
    5. Account does not have a patient or self-pay balance (AccountBillingCycle)
    6. All of the Account's linked accounts (mom-baby and pre-surgery/surgery only) are purge-eligible.
2. During a purge job, purge order is based on account age (Account.AccountCreatedDate) from oldest to newest purge-eligible record.
3. Purge removes the oldest purge-eligible parent accounts as well as all linked child records in batches of 30000 (*PurgeAccountBatchSize*) that meet the *MinimumAccountLastZeroedDaysConstraint* property. Oldest accounts, based on Account.CreatedDate, are purged first, as well as their child records.
4. Purging is logged in Control.dbo.PurgeHistory and Control.dbo.PurgeAccountHistoryDetail
5. Orphaned records in MedicalRecord are purged. This includes MedicalRecord undefined records.

6. Orphaned Guarantor records are purged.
7. Orphaned WorkItemGroup rows are purged.
8. Orphaned Accounts in BatchImport.dbo.AccountLookup are purged.
9. Orphaned Incidents in MessageDB.PackageCache table are purged.
10. Includes a check for the existence of the AccountClinician table. If it exists, purge from the table.
11. The PurgeAccounts sproc removes purge-eligible parent accounts and their child records for the current batch by table in the following order; all rows for all accounts in the current batch are deleted as a group:

1	WorkItemRelationship	18	Bill
2	WorkItemGroupEligibility	19	Denial
3	WorkItemDomainList	20	AccountClinician
4	TrackedWorkItemActivity	21	AccountProvider
5	ActivityDetail	22	AccountBillingError
6	Activity	23	AccountDeficiency
7	TrackedTaskActivity	24	BillingCycleSponsorPlan
8	Task	25	AccountSponsorPlan
9	DomainModuleOutcome	26	AccountTransaction
10	Escalation	27	AccountBillingCycle
11	WorkItem	28	Visit
12	Incident	29	Account
13	VisitProcedureType		Orphans - removed once at the end, not in batches:
14	VisitProvider		Guarantor
15	PaymentVariance		Medical Record
16	ChargeItem		WorkItemGroup
17	BillChargeItem		BatchImport.dbo.AccountLookup
			MessageDB.dbo.PackageCache

12. Purge stops for any of the following reasons:
    - All purge-eligible parent accounts as well as all linked child records are purged.
    - The configured number of batches (*PurgeAccountNumberOfBatches*) has been processed.
    - The purge time frame (*PurgeAccountRunTime*) is exceeded.

      At the end of the purge time frame, the PurgeAccounts sproc processes the batch in progress to completion. After that, no more batches are started even if the *PurgeAccountNumberOfBatches* has not been reached.
    - An error that that stops the batch deletion process has occurred.

      By default, the *PurgeAccountMaxBatchesWithError* is 1. For example, if the purge encounters an error while processing the first batch deletion, it logs the error, rolls back any changes, and goes on to the next batch. If it finds another error in the second batch, it logs the error, rolls back any changes, and stops the entire purge process.

      If necessary, you can modify the system not to skip errors at all by changing *PurgeAccountMaxBatchesWithError* to 0. In this case, if the purge encounters an error while removing records, it logs the error, rolls back any changes, and stops.

Parent/Sub-Parent Record Relationships

Sub-Parent	Parent(s)
Account Billing Error	Account

Account Clinician	Account
Account Deficiency	Account
Account Provider	Account
AccountBillingCycle	Account
AccountSponsorPlan	Account
AccountTransaction	Account
BillingCycleSponsorPlan	Account
ChargeItem	Account
Denial	Account
PaymentVariance	Account
TrackedTaskActivity	Account
TrackedWorkItemActivity	Account
Account	AccountID
Bill	AccountID
Incident	AccountID
Visit	AccountID
ActivityDetail	ActivityID
Activity	ActivityID via IncidentID via AccountID
BillChargeItem	Bill
WorkItem	IncidentID via AccountID
VisitProcedureType	Visit
VisitProvider	Visit
DomainModuleOutcome	Visit, Workitem
Escalation	WorkItem
Task	WorkItem
WorkItemGroupEligibility	WorkItem
WorkItemDomainList	WorkItem, AccountBillingCycleID, VisitID
WorkItemRelationship	WorkItemID - Left/RightWorkItemId
BatchImport.AccountLookup	AccountSourceIdentifier
MessageDB.PackageCache	IncidentID

See Appendix A: HealthcareWorkManagement Tables Not Affected by Any TRX Purge for HealthcareWorkManagement tables not affected by purge.

ABC Purge Process

Account Billing Cycle (ABC) Purge deletes purge-eligible AccountBillingCycle and its child records whether the account and other related records do or do not qualify for Account Purge. ABC Purge is driven from the same PurgeConfigSetting table and follows the same basic logic and workflow as Account Purge.

A SQL Server Agent job called TransactionalABCPurge runs the PurgeAccountBillingCycle sproc at a scheduled interval. By default, it runs weekly at 10:00 AM on Saturday, after the Account Purge.

1. The PurgeAccountBillingCycle sproc identifies purge-eligible records at the beginning of the run (all numbers used are default values of *PurgeConfigSetting* items-shown as *green italics*) based on the following properties of the parent account billing cycle. All the following must be true for an account billing cycle to be purge-eligible:

- The account billing cycle cannot be an accumulator cycle (IsAccumulator = 0).
- The cycle must have a zero payer and patient balance for 730 consecutive days (*AccountBillingCycleEndDate)* based on the cycle end date.
- All the cycle's back-end action items (BL, CO, SP, RQ) are closed for 180 consecutive days (*AccountBillingCycleWorkItemClosedDays*) and there has been no activity on these items for 60 consecutive days (*AccountBillingCycleLastTouched*). The action item closed criteria is based on the WorkItem.WorkItemState and WorkItem.DateCompleted fields.
- A null payer or patient balance is treated as zero because a null indicates it never had a balance. If an account billing cycle has no activities it qualifies. If a workitem is in a closed state with a null DateCompleted, it meets the time criteria to qualify. However, a null cycle end date does not qualify since this would work effectively like epoch date.

All purge-eligible billing cycles and only their back-end action item types are purged.

2. During a purge job, purge order is based on CycleEndDate. Purge removes the oldest purge-eligible parent account billing cycles as well as all linked child records in batches of 30000 (*PurgeABCBatchSize*).
3. Purging is logged in Control.dbo.PurgeHistory and Control.dbo.PurgeABCHistoryDetail.
4. The PurgeAccountBillingCycle sproc removes purge-eligible parent account billing cycles and their child records for the current batch by table in the following order; all rows for all account billing cycles in the current batch are deleted as a group:

1	WorkItemRelationship LeftKey	12	WorkItem
2	WorkItemRelationship RightKey	13	PaymentVariance
3	WorkItemGroupEligibility	14	ChargeItem
4	WorkItemDomainList delete by AccountBillingCycleID	15	BillChargeItem
5	TrackedWorkItemActivity	16	Bill
6	ActivityDetail	17	Denial
7	Activity	18	AccountBillingError
8	TrackedTaskActivity	19	AccountDeficiency
9	Task	20	BillingCycleSponsorPlan
10	DomainModuleOutcome via WorkItem	21	AccountTransaction
11	Escalation	22	AccountBillingCycle

5. Purge stops for any of the following reasons:
   - All purge-eligible parent account billing cycle records as well as all linked child records are purged.
   - The configured number of batches (*PurgeABCNumberOfBatches*) has been processed.
   - The purge time frame (*PurgeABCRunTime*) is exceeded.

     At the end of the purge time frame, the PurgeAccountBillingCycle sproc processes the batch in progress to completion. After that, no more batches are started even if the *PurgeABCNumberOfBatches* has not been reached.
   - An error that that stops the batch deletion process has occurred.

     By default, the *PurgeAccountMaxBatchesWithError* is 1. For example, if the purge encounters an error while processing the first batch deletion, it logs the error, rolls back any changes, and goes on to the next batch. If it finds another error in the second batch, it logs the error, rolls back any changes, and stops the entire purge process.

If necessary, you can modify the system not to skip errors at all by changing *PurgeAccountMaxBatchesWithError* to 0. In this case, if the purge encounters an error while removing records, it logs the error, rolls back any changes, and stops.

Parent-SubParent Record Relationships

Sub-Parents	Parent(s)
WorkItem	From WorkItemDomainList via AccountBillingCycleID
Task	From ActivityDetail via WorkItemID
Activity	From ActivityDetail via WorkItemID Also from ActivityDetail via TaskID
BillingCycleSponsorPlan	From BillingCycleSponsorPlan via AccountBillingCycleID
Bill	From Bill via BillingCycleSponsorPlanID
ChargeItem	From BillChargeItem via BillID

General Process Points

- Temp tables are populated with the IDs (primary keys) of the rows in parent tables that are to be deleted. All the tables to be purged can be accessed using one of these parent/sub-parent rows. Each type of sub-parent row is associated with the corresponding parent AccountID or AccountBillingCycleID which is at the root of that chain. For example, AccountID can be used to access most tables; however, some tables need to be accessed via other keys from children of Account, such as VisitID or WorkItemID.
- There is no way to turn purge on or off for any individual table. All involved tables are purged.
- As much processing as is reasonable is kept out of the transaction that is doing the actual delete.
- Errors are logged in the Control database in the PurgeHistory and PurgeCandidate Tables in the RunMessage column, and in the ExceptionLog table.
- Schedule purge to run while the system is quiet. It should not be run during import or elapsing day.
- **Identify only mode:** a test mode that is controlled by the *PurgeAccountIdentifyOnlyMode* or *PurgeABCIdentifyOnlyMode* setting.
    - This mode identifies accounts that would have been purged but does not delete them.
    - The purge run is marked as a test only run in PurgeHistory.RunMessage.
- **Verbose Mode:** use the *PurgeAccountVerboseMode* or *PurgeABCVerboseMode* setting to enable Verbose mode for troubleshooting purposes.

  This mode works only when you run the PurgeAccounts or PurgeAccountBillingCycle sproc directly in SSMS because the resulting messages are displayed on the Output tab in SSMS. It outputs the number and duration of row deletions from each table per batch, as well as the number of purged orphans. It does not work when you use Verbose mode running the sproc from the SQL Agent job.

  Default for Verbose mode is “Lean”, no messages.
- Although you can set the purge sproc with both Verbose mode and Identify mode on at the same time, you will not get any messages. You must be in Purge mode, not Identify mode, to get messages from Verbose mode (messages are only created with actual record deletion).

Tables

Control.dbo.PurgeConfigSetting

This table stores all the settings for the TRX Purge. Each row in this table is a configuration setting. For more detail, see Purge Configuration Settings Table.

Table columns: PurgeConfigSetting

PurgeConfigSettingID	INT IDENTITY(1,1)
SettingName	VARCHAR(75)
SettingValueINT	INT
SettingValueVChar	VARCHAR(100)
SettingEffectiveDate	DATETIME
SettingInactiveDate	DATETIME
SettingNotes	VARCHAR(2000)

All settings have a value in either SettingValueINT or SettingValueVChar but not both.

A trigger enforces business rules (e.g., prevent invalid values and date range overlaps).

Control.dbo.PurgeCandidate

During first part of the purge process, which is the purge candidate identification phase, this table is loaded with the IDs of purge-eligible candidate parents and some sub-parent rows (not every child table's primary keys need to be inserted into this table) from the ReportingDW database. During the second part of the process, these identified records are removed from ReportingDW in batches.

Table Columns: PurgeCandidate

ParentID	INT
ParentCreateDate	INT
PKTableSource	VARCHAR(30)
PKID	INT
EligibilityDate	Date
PurgeHistoryID	INT
BatchNumber	INT
RunMessage	Varchar(2000)

Table Constraints

```
ALTER TABLE PurgeCandidate
  ADD CONSTRAINT FK_PurgeCandidate_To_PurgeHist FOREIGN KEY (PurgeHistoryID)
REFERENCES PurgeHistory(PurgeHistoryID)
```

How PurgeCandidate Works

This table remains populated between purge runs, allowing you to analyze and troubleshoot the purge results. At the beginning of the next purge run, all rows for records identified as purge-eligible from the previous run are removed, even if they were not purged because of the time constraint. The exception records that failed purge (for error conditions) are retained for period of time for analysis. PurgeCandidate is a working table; it is not a history or logging table.

PurgeCandidate has one row per parent account or parent account billing cycle that is a candidate for purge. Rows are inserted at the beginning of a purge run along with a date and the ID of the corresponding PurgeHistoryID.

This table also holds the sub-parent rows (e.g., VisitID, WorkItemID, or TaskID) needed to identify tables that cannot be accessed directly via the ParentIDs (AccountID or AccountBillingCycleID).

Once a parent is selected to be purged during a run as part of a batch, the process purges that set of records and then updates the candidate table as follows.

- At the beginning of each batch, PurgeCandidate rows are updated with the batch number during which they are selected to be purged.
- Parent and sub-parent rows records where purge is attempted but failed are updated with the associated failure messages for the batch that was running when the error occurred. These rows are retained for the period defined by the *PurgeAccountCandidateErrorRetentionDays* value. The records outside the retention window are removed during the next purge run.
- Successfully purged parent and sub-parent rows are removed from the PurgeCandidate table during the next run of the process. Each purge (account or ABC) only removes its own rows.

Except for retained parents, PurgeCandidate rows are deleted at the beginning of each run.

At the end of a purge run, this table is in one of these states:

- Empty for this purge type: no purge-eligible records found, or this purge has never run.
- Has rows with all identified records that were successfully purged (to be removed at the beginning of the next run). BatchNumber is populated and RunMessage is null = successful run. PurgeHistory table for that run should also reflect success. Based on PurgeHistoryID.
- Has rows with null Batch Number: run ended before attempting to purge this parent due to time constraint (to be removed at the beginning of the next run).
- Has rows with non-null Batch Number AND non-null Run Message: purge attempted but failed for this set of parents. Failure info is in RunMessage column. These are retained for a configured number of days and then deleted (*PurgeAccountCandidateErrorRetentionDays*).

Control.dbo.PurgeHistory

This table is a historical log where the PurgeAccounts stored procedure adds a row for every run of the purge. The account level detail that links to each record is in the PurgeAccountHistoryDetail table.

Table columns: PurgeHistory

PurgeHistoryID	INT IDENTITY(1,1)
StartDateTime	DATETIME
EndDateTime	DATETIME
RunStatus	Varchar(10)
NumberOfPurgedParents	INT
PurgeBatchSize	INT

NumberOfBatches	INT
PurgeType	Varchar(30)
RunMessage	Varchar(2000)
TargetDatabase	Varchar(30)

Table constraints

```
ALTER TABLE PurgeHistory
  ADD CONSTRAINT PK_PurgeHistory PRIMARY KEY CLUSTERED (PurgeHistoryID)
```

Control.dbo.PurgeAccountHistoryDetail

Each account that is purged in an account purge run has a row in this table. The purge run information is linked in the PurgeHistory table.

Table columns: PurgeAccountHistoryDetail

PurgeAccountHistoryDetailID	INT IDENTITY(1,1)
PurgeHistoryID	INT
AccountSourceIdentifier	NVARCHAR(75)
AccountID	INT
AccountCreatedDate	DATE
BatchNumber	INT

Table constraints

```
ALTER TABLE PurgeAccountHistoryDetail
  ADD CONSTRAINT PK_PurgeAccountHistoryDetail PRIMARY KEY CLUSTERED
(PurgeAccountHistoryDetailID)

ALTER TABLE PurgeAccountHistoryDetail
  ADD CONSTRAINT FK_AcctPurgeDetail_To_PurgeHist FOREIGN KEY (PurgeHistoryID)
REFERENCES PurgeHistory(PurgeHistoryID)
```

For release version 14.4.3, accounts are only the only purge parent type. In the future, account billing cycles will be added as a parent type.

Control.dbo.PurgeABCHistoryDetail

Each AccountBillingCycle that is purged in an AccountBillingCycle purge has a row in this table.

Table Columns: PurgeABCHistoryDetail

PurgeABCHistoryDetailID	INT IDENTITY(1,1)
PurgeHistoryID	INT
AccountBillingCycleID	INT
AccountID	INT
CycleStartDate	DATETIME2(0)
BatchNumber	INT

Table constraints

```
ALTER TABLE PurgeABCHistoryDetail
  ADD CONSTRAINT PK_PurgeABCHistoryDetail PRIMARY KEY CLUSTERED
(PurgeABCHistoryDetailID)

ALTER TABLE PurgeABCHistoryDetail
  ADD CONSTRAINT FK_PurgeABCDetail_To_PurgeHist FOREIGN KEY (PurgeHistoryID)
REFERENCES PurgeHistory(PurgeHistoryID)
```

Control.dbo ExceptionLog

This is a new general table of which TRX Purge is the first usage. The PurgeAccounts sproc logs purge errors to this table.

Table Columns: ExceptionLog

ExceptionID	bigint Identity(1,1) NOT NULL,	
ExceptionSource	varchar(256) NOT NULL,	(Ex: "Purge")
SourceIdentifier	varchar(256) NULL,	(Ex: "AccountSourceIdentifier/VisitSourceIdentifier")
ExceptionMessage	varchar(4096) NOT NULL,	
ExceptionType	varchar(100) NULL,	(For Purge/SQL – null; for .NET – exception type)
ExceptionCode	varchar(20) NULL,	(For Purge - Error number)
ExceptionSeverity	int NULL,	(Information - 0, Warning - 1, Error - 2, Severe - 3, Critical - 4) - For purge use 2 for all the errors.
ExceptionScope	varchar(256) NULL,	(purge - null)
ExceptionInstanceID	varchar(38) NULL,	(Purge - null)
ServiceName	varchar(256) NULL,	(Purge - null)
MachineName	varchar(256) NULL,	
ExceptionDateTime	datetime NULL,	
ExceptionSourceProcedure	varchar(256) NULL,	(Purge - SQL Procedure name that caused the error)
ExceptionStackTrace	varchar(4096) NULL,	(Purge null)
InnerExceptionMessage	varchar(4096) NULL,	(Purge null)
ExceptionLineNumber	int NULL,	(Purge - Stored proc line number)
SQLExceptionSeverity	int NULL,	(Purge - SQL Error Severity)
SQLExceptionState	int NULL,	(Purge - SQL State)

Modifying TRX Purge Configuration

Configure the TRX Purge by either modifying its SQL Server Agent job or configuration table in the Control database on the Aeos database server.

Schedule

You can adjust when the TRX Purge runs by changing the schedule of the SQL Server Agent jobs.

- For Account Purge, edit the schedule of the TransactionalPurge job. It runs the HealthcareWorkmanagement.dbo.PurgeAccounts stored procedure.
- For Account Billing Cycle Purge, edit the schedule of the TransactionalABCPurge, which runs the PurgeAccountBillingCycle stored procedure.

Schedule the purges to run during system quiet times. For example, during the weekend, with no imports, user activity, or other database maintenance processes.

By default, the purge jobs are disabled. Enable the jobs to turn them on.

Purge Configuration Settings Table

All configuration settings are stored in the Control.dbo.PurgeConfigSetting table. Change a configuration for a particular setting by inserting a row into to table usually with the new SettingValue (SettingValueINT or SettingValueVChar), SettingEffectiveDate, and SettingNotes, leaving the rest of the values the same as its previous setting.

Table Columns: PurgeConfigSetting

Field Name	Type	Description
PurgeConfigSettingID	INT IDENTITY(1,1)	Table primary key
SettingName	VARCHAR(75)	Name of the configuration setting. There can be multiple entries of a SettingName for different date ranges and associated SettingValues.
*SettingValueINT	INT	Integer setting value. If this column is not NULL, then the SettingValueVCHAR column must be NULL.
*SettingValueVChar	VARCHAR(100)	Variable character setting value If this column is not NULL, then the SettingValueINT column must be NULL.
*SettingEffectiveDate	DATETIME	Beginning of the date/time range for a value. Date/time ranges for a setting cannot overlap nor have gaps. If you add an overlapping range, the old ranges are adjusted to accommodate the new change.
SettingInactiveDate	DATETIME	End of the date range for a value.
*SettingNotes	VARCHAR(2000)	Description of the setting; you can also use this field to keep implementation notes or comments.

*In most cases, only these fields are modified during reconfiguration.

Table Rows: PurgeConfigSetting

This table has the settings for Account Purge and ABC Purge.

Setting defaults and notes from the PurgeConfigSetting table:

SettingName	Default	SettingNotes
Account Purge Settings		
AccountLastZeroedWithDenials	365	Maximum age in days that an account with denials has had a zero balance before it is purge-eligible. Calculated from the HCWM.Account.AccountBalanceZeroedDate. Minimum value: 180

SettingName	Default	SettingNotes
AccountLastZeroedWithoutDenials	180	Maximum age in days that an account without denials has had a zero balance before it is purge-eligible. Calculated from the HCWM.Account.AccountBalanceZeroedDate. Min value:180
ActionLastTouchedWithDenials	365	Maximum age in days that an action item on an account with denials was last touched before it is purge-eligible. Calculated from the HCWM.Activity.ActivityDateTime. Min value: 180
ActionLastTouchedWithoutDenials	180	Maximum age in days that an action item on an account without denials was last touched before it is purge-eligible. Calculated from the HCWM. Activity.ActivityDateTime. Min value: 180
MinimumAccountLastZeroedDaysConstraint	180	Maximum age in days that an account last went to zero balance before it is purge-eligible. Regardless of whether or not the account has denials. Used for enforcing constraint on other config values and to drive initial selection of population. Calculated from the HCWM.Account.AccountBalanceZeroedDate. Min value: 180 Use the lowest value of the following settings: AccountLastZeroedWithDenials, AccountLastZeroedWithoutDenials, ActionLastTouchedWithDenials, ActionLastTouchedWithoutDenials **NOTE:** This setting is removed from Aeos new stack versions 15.2 and later.
PurgeAccountBatchSize	30000	How many parent Account Id values of purge-eligible records are selected for each batch in PurgeAccount. Recommended value: 30000
PurgeAccountIdentifyOnlyMode	Identify	Set PurgeAccount to identify accounts but not actually purge them, or to identify AND purge them. Valid values: Identify, Purge
PurgeAccountNumberOfBatches	50	How many batches to process during execution. Min value: 1
PurgeAccountRunTime	60	Number of minutes the process runs before it stops selecting additional batches to process. It finishes up the current batch, skips the remaining batches, and ends with Success. No minimum
PurgeAccountVerboseMode	Lean	Sets PurgeAccount to produce output messages for QA and debug or not. Valid values: Verbose, Lean
Account Billing Cycle Purge Settings		
AccountBillingCycleEndDate	730	Maximum age in days of an account billing cycle before it is purge-eligible. Calculated from HCWM.Account.AccountBalanceCycle.PatientBalance and HCWM.BillingCycleSponsorPlan.PayerBalance and used ABC.CycleEndDate. Min value:180
AccountBillingCycleLastTouched	60	Maximum age in days that an action item on an AccountBillingCycle was last touched before it is purge-eligible. Calculated from the HCWM.Activity.ActivityDateTime. Min value: 60
AccountBillingCycleWorkItemClosedDays	180	Maximum age in days ago that a WorkItem associated with an AccountBillingCycle was closed before it is purge eligible. Calculated from the HCWM.dbo.WorkItem.DateCompleted. Min value: 180
PurgeABCBatchSize	30000	How many AccountBillingCycleId values are selected as parents for each batch in PurgeAccountBillingCycle. Recommended value: 30,000
PurgeABCIdentifyOnlyMode	Identify	Set PurgeAccountBillingCycle to identify accounts but not actually purge them, or to identify AND purge them. Valid values: Identify, Purge
PurgeABCNumberOfBatches	50	How many batches to process during execution of ABC Purge. Min value: 1
PurgeABCRunTime	60	Number of minutes the process runs before it stops selecting additional batches to process. It finishes up the current batch, skips the remaining batches, and ends with Success. No minimum
PurgeABCVerboseMode	Lean	Sets PurgeAccountBillingCycle to produce output messages for QA and debug or not. Valid values: Verbose, Lean

SettingName	Default	SettingNotes
Shared Settings		
PurgeAccountCandidateErrorRetentionDays	45	Duration in days that a row in PurgeCandidate with an error is retained before being deleted by the process. Min value: 1 Used by both Account and ABC Purges.
PurgeAccountMaxBatchesWithError	1	Number of batches that can have an error without stopping purge. Default of 1 means that one batch can have an error but execution stops at two. Min value: 0 Used by both Account and ABC Purges.

The settings for TRX Purge can be categorized as follows. Settings apply to either account or account billing cycle parent records as named in their respective purge processes, except for the max batches and error retention settings, which apply to both. SettingName is displayed in *green italics*.

- **Purge eligibility–**specify age criteria that determines which parent records can be purged using the following fields:

  For Account Purge: *AccountLastZeroedWithDenials*, *AccountLastZeroedWithoutDenials*, *ActionLastTouchedWithDenials*, *ActionLastTouchedWithoutDenials*, and **MinimumAccountLastZeroedDaysConstraint* (NOTE: *MinimumAccountL`astZeroedDaysConstraint* is removed from new stack-versions 15.2 and later).

  For ABC Purge: *AccountBillingCycleEndDate*, *AccountBillingCycleLastTouched*, and *AccountBillingCycleWorkItemClosedDays*.

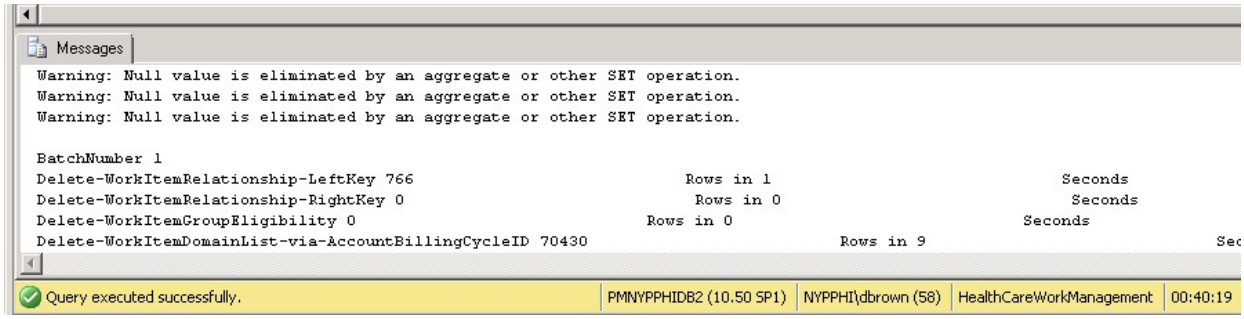
  If you have a large amount of data to purge that has built up over time, you may want to move these dates further into the past, such as when you first start running this process at a particular client site.

- **Number**–Specify how many of the eligible parent (and all associated child) records to delete during a purge run using *PurgeAccountBatchSize, PurgeAccountNumberOfBatches, PurgeABCBatchSize,* and *PurgeABCNumberOfBatches*.
  - This number specifies purge-eligible parent records only; associated child records are not included in this number but are deleted along with their parent records.
  - Larger batch sizes are more efficient due to unavoidable overhead per batch and per statement.
  - Unless you are experiencing performance issues, you likely do not need to change these settings. Contact the DevOps or Product Management team if this setting needs to be changed.
- **Time limit**–control how long the purge job can run using *PurgeAccountRunTime* and *PurgeABCRunTime*.
  - This is the time after which the process will not start another batch.
  - This does not indicate the max run time of the process.
  - This is the best way to control the approximate run time of your Purge Job.
  - Be aware that the job will run a bit longer than this setting as it needs to finish the current batch and process orphans as well as final processing steps before it will complete.
- **Error tolerance**–limit the number of errors that are allowed before stopping the purge sproc using *PurgeAccountMaxBatchesWithError*. This setting is shared by both purges.
- **Error retention**–-limit number of days that PurgeCandidate table rows are retained for parent records where purge was attempted but failed using *PurgeAccountCandidateErrorRetentionDays*. This setting is shared by both purges.
- **Verbose mode**–provide more detail for QA and debugging using *ModePurgeAccountVerboseMode* and *PurgeABCVerboseMode*.

Valid values: Verbose, Lean.

- Verbose mode - produce output messages for QA and debugging.
- Lean (default) - no output messages.

These details are output to the Messages tab in SSMS only when the PurgeAccounts or PurgeAccountBillingCycle sproc is run manually (not in the SQL Server Agent Job). Copy this output to an Excel document to save it for analysis (See TRXAccountPurgeVerboseOutputExample.xlsx).



If you run in verbose mode from the SQL Server Agent job, the purge runs slower and it is difficult to locate the messages via SQL Server logs or they may not be captured at all.

- **Identify only mode**–identify purge-eligible parent records without removing them using *PurgeAccountIdentifyOnlyMode*. In this case, the PurgeCandidate table rows retain all purge-eligible parent records without purging any data. These parents are removed from PurgeCandidate during the next purge run.

  Valid values: Identify, Purge.

  - Identify mode (default) - identify purge-eligible records, but do not actually remove them.
  - Purge - identify and remove purge-eligible records.

Initial Analysis/Set Up

When you implement Account Purge, you need to set up the configuration differently for an initial period while it processes a system with a large amount of old and unnecessary data. Once it clears that initial backlog, it can run at its regular schedule, usually with the default settings.

Before Configuring

Identify the oldest record by selecting the minimum account created date.

```
DECLARE @OldestAcctDate DATE = (SELECT MIN(a.CreatedDate) FROM Account A)
```

How many days old is it?

```
SELECT DATEDIFF(dd,@OldestAcctDate,GETDATE())
```

Strategies for Initial Account Purge Runs

When you first implement account purge, you will likely need to work through a large backlog of purge-eligible records. There are two approaches, depending on whether you can work through the backlog all at once or need to catch up over time (weekly).

All At Once

You can do a large initial purge run if you can get sufficient time and resource allocated for purging through the initial backlog (for example if you have secured an 8 hour time slot over a weekend).

Change the following settings in the Control.dbo.PurgeConfigSetting table so that you can maximize your initial purge-eligible candidate record identification phase. It will run longer since you are putting more data into the hopper, but will not need to run as many iterations of the job and there will be more data to process during batches:

- Insert a new row for the setting *PurgeAccountRunTime* using a value an hour less than your window. Remember this setting is in minutes. Since you are running a large set, this leaves plenty of time for post-batch processing to occur and ensures completion within the 8 hour window.
- Insert a new row for the setting *PurgeAccountNumberOfBatches* using a high number. You want it to stop processing based on time, not on number of batches, so make it something like 1000.
- Make sure *PurgeAccountVerboseMode* is set to **Lean** for large purge runs.
- Make sure *PurgeAccountIdentifyOnlyMode* is set to **Purge**.

Catch Up (Weekly)

Due to time/resource limitations, you may need to slowly catch up to your data goals by purging the through the initial backlog of eligible records over time. This approach assumes that you will be running TRX purge on its intended maintenance schedule of once per week during a quiet time.

1. For your first run, start at 100 days past the oldest record.
   - *AccountLastZeroedWithDenials* = OldestAcctDate + 100 (100 days later)
   - *AccountLastZeroedWithoutDenials* = OldestAcctDate + 100
   - *MinimumAccountLastZeroedDaysConstraint* = OldestAcctDate + 100

   For example, with the oldest account at 706 days old, you can insert these new settings into the PurgeConfigSettings table:

```
INSERT INTO Control.DBO.PurgeConfigSetting
  (SettingName, SettingValueINT, SettingEffectiveDate)
VALUES
  ('AccountLastZeroedWithDenials', 606, '01/13/2015')

INSERT INTO Control.DBO.PurgeConfigSetting
  (SettingName, SettingValueINT, SettingEffectiveDate)
VALUES
  ('AccountLastZeroedWithoutDenials', 606, '01/13/2015')

INSERT INTO Control.DBO.PurgeConfigSetting
  (SettingName, SettingValueINT, SettingEffectiveDate)
VALUES
('MinimumAccountLastZeroedDaysConstraint', 606, '01/13/2015')        )
```

   - If one of these date calculations is newer than a business constraint (minimum value in the Table Rows: PurgeConfigSettings section), use the minimum setting. A trigger prevents insertion of a value outside the constraints.
   - Minimum values set by the business are included in the SettingNotes column for each row for your reference.
   - Leave *PurgeAccountRunTime* and other settings at their default values.

2. Run with these settings according to the TransactionalPurge job schedule until the duration of the most recent purge is less than the allotted time (*PurgeAccountRunTime*); this means that all purge-eligible rows are removed.

   To see how many candidates were identified but not purged during the most *recent* run:

```
select COUNT(1)
  from PurgeHistory ph
  join PurgeCandidate pc
    on ph.PurgeHistoryID = pc.PurgeHistoryID
 where pc.PKTableSource = 'Account'
   and pc.BatchNumber is null
```

3. When all purge-eligible records can be removed in a single purge run, move the date another 100 days forward by adding 100 to *AccountLastZeroedWithDenials*, *AccountLastZeroedWithoutDenials*, and *MinimumAccountLastZeroedDaysConstraint*.

```
INSERT INTO Control.DBO.PurgeConfigSetting
  (SettingName, SettingValueINT, SettingEffectiveDate)
VALUES
  ('AccountLastZeroedWithDenials', 506, '01/13/2015')

INSERT INTO Control.DBO.PurgeConfigSetting
  (SettingName, SettingValueINT, SettingEffectiveDate)
VALUES
  ('AccountLastZeroedWithoutDenials', 506, '01/13/2015')

INSERT INTO Control.DBO.PurgeConfigSetting
  (SettingName, SettingValueINT, SettingEffectiveDate)
VALUES
('MinimumAccountLastZeroedDaysConstraint', 506, '01/13/2015')        )
```

4. Again, run with these settings until the purge removes all the purge-eligible rows, as indicated by the most recent run completing in less than an hour (configured value of PurgeAccountRunTime)

5. Repeat this process until you are able to step back to the default settings and keep those values going forward.

Strategies for Initial ABC Purge Runs

Due to the 730 day default value for the *AccountBillingCycleLastZeroed* setting, most clients will not have data to purge initially. If you want to change this setting closer to its minimum value of 180, you should carefully analyze the client's data profile beforehand.

Troubleshooting

Job/Stored Procedure Will Not Run

Reason 1: Another purge process is already running on the HCWM database.

Make sure that the other purge process is not running.

Reason 2: If the PurgeAccounts or PurgeAccountBillingCycle sproc was stopped before catch block (e.g., SPID killed) it cannot be run again because the Control.dbo.PurgeHistory.RunStatus is "Running". It could also not start because there is another purge process running in that database.

You can update the row for that value of PurgeHistoryID to have a RunStatus of anything other than "Running" such as "ManualEnd". The job will start again as long as the most recent row is not in "Running" status.

Appendix A: HealthcareWorkManagement Tables Not Affected by Any TRX Purge

The following tables in HealthcareWorkManagement are not affected by the purge processes.

__RefactorLog	ActivityType	AdmitSourceType	AdmitStatusType
AppUser	AtRiskStatusType	BaseTaskOutcome	BaseTaskType
BaseWorkItemType	BillingErrorType	Calendar	CalendarConfig
CashPoster	Clinician	ClinicianType	CollectionGroup
Collector	DataType	DeficiencyType	DenialType
DiagnosisType	DisplayElement	ErrorCategory	ErrorSource
EscalationOutcome	EscalationStatus	EscalationType	FilterColumn
FilterColumnValue	FilterControlType	FilterOperator	FilterOperatorControl
FinancialClassType	Hierarchy	HierarchyLevel	HierarchyNode
HuronVersion	ICDCodeMap	IncidentType	LanguageType
LevelType	MajorDenialType	MajorPatientType	MinorDenialType
MinorPatientType	Module	Node	NodeType
OrganizationalNodePatientTyp e	OrganizationalNodeSponsorPla n	OutcomeType	PatientType
Payer	PayerGroup	PayerType	PaymentVarianceType
PendReason	Phantom_Activity_WIS	Phase	PortfolioFilter
PortfolioFilterDetail	PortfolioFilterGroup	PortfolioFilterHistory	PortfolioFilterType
ProcedureType	PropensitySegmentCategory	Provider	ProviderClinician
ReEvaluationSource	ScheduledStatusType	SelfPayPlan	SelfPayPlanCategory
ServiceType	SponsorPlan	Strata	StrataFamily
StrataRange	SubordinateValueType	TaskStatus	TaskStatusCategory
TaskStatusType	TaskType	Team	TeamDetailDisplayElement
TeamDisplayElement	TeamFilterColumn	TeamGridDisplayElemen t	TeamOrganizationalNodeFilte r
TeamRepFilter	TeamRepFilterDetail	TeamRepFilterType	TeamRepresentative
TeamScoreThreshold	TimingStrataGroup	TransactionCategory	TransactionCode
TransactionGroup	TransactionType	Vendor	WorkItemGroupType
WorkItemLinkType	WorkItemStamp	WorkItemStampTask	WorkItemState
WorkItemStatus	WorkItemStatusCategory	WorkItemSubType	WorkItemType
WorkItemTypeSubTypeLookup			